PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	
Neil W. Black, et al.	Atty. Dkt. No.: 003797.86776
Appln. No.: 09/592,660	Group Art Unit: 3628
Filed: June 13, 2000	Examiner: F. Poinvil
For: SYSTEM AND METHOD FOR CREATING MODEL INVESTMENT PORTFOLIOS	

DECLARATION UNDER 37 C.F.R. § 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

I, Peter Hansen, do hereby declare as follows:

1. I am named as an inventor in U.S. Patent Appln. No. 09/592,660, filed June 13, 2000, along with Neil W. Black and Jon D. Markman. I am an employee of Microsoft Corporation of Redmond, Washington, and I am over the age of eighteen years.

2. I am advised that Exhibit A contains a true and correct copy of the present pending claims in U.S. Patent Appln. No. 09/592,660, namely claims 1-11, 17-28, and 30-48. I have read and understand these claims.

3. I also have been informed that the United States Patent and Trademark Office has rejected claims 1-11, 17-28, and 30-48 as contained in Exhibit A based on U.S. Patent No. 6,493,681 to Tertitski, et al. I understand that the Tertitski patent claims an effective U.S. filing date of August 11, 1999.

4. I reviewed computer code entitled "setupwiz" (hard copy attached as Exhibit B) and "setupwiz.h" (hard copy attached as Exhibit C). I was the original author of this code. I understand that these copies of the code were obtained by my co-inventor, Neil Black, from our company's records. The copy of "setupwiz" attached as Exhibit B has been changed from that included with Neil Black's Declaration to include line numbers for the code lines (the file, including the line numbers, was renamed "setupwizlines" in Exhibit B). The line numbers were added in Exhibit B to make the discussion below easier to follow, *i.e.*, so that the cited code line numbers below would be readily available with the attached copy of the code. The addition of line numbers is the only difference between this copy of the "setupwiz" code and the copy provided by Neil Black.

5. As will be described in more detail below, the attached computer code, when implemented on a computer system, allowed one to create a model portfolio of stock issues in the manner described in our patent application and its claims (Exhibit A).

6. One feature of the claims in our patent application relates to "receiving computer input identifying specific issues to be added to a portfolio" (note, for example, claim 1, line 3; claim 17, line 4; claim 28, line 3; claim 30, line 4; claim 34, line 3; claim 38, line 3; claim 43, line 4; and claim 48, line 2). Line 3066 of the "setupwiz" code (page 56) initiates a call to "PropertySheet," which launches the New Account wizard for the portfolio manager. A list of pages potentially displayable in the method "InitAccountWizardPSP( ) may be found beginning at line 2762 of the "setupwiz" code (page 51). The page pertaining to model portfolios is commented "Watch Accounts" at line 2791 (page 51).

7. The "PropertySheet" call fills in a "SETUP_STRUCT" structure with the various options selected by a user. The user can enter a list of stock tickers in the edit box of a dialog box (*e.g.*, see boxes 403 and 504 from Figs. 4 and 5, respectively, of our patent application), and those tickers are passed into the method "SetupWatchAccount( )" referenced on line 2609 of the setupwiz code (page 48). The input stock tickers are validated as corresponding to actual securities when the call to "LookupPrices( )" is made at line 2655 (page 49). The "setupwiz.h" code (Exhibit C) includes the definitions for the "SETUP_STRUCT" structure.

8. The code for actually implementing the business logic behind the controls on the model portfolio page dialog includes the contents of "AccountWizWatchDialogProc," which begins at line 579 of the "setupwiz" code (page 11).

9. Another feature of several claims in our patent application relates to "receiving computer input indicating a selection of an option for creating the portfolio" (note, for example, claim 1, lines 4-5; claims 2-10; claim 17, line 5; claims 18-26; claim 28, lines 3-4; claim 30, line 7; claim 34, lines 7-8; claim 38, lines 4-5; claims 39-41; claim 43, line 5; claims 44-46; and claim 48, lines 2-3).

10. In the "New Account Wizard," users have the option of checking a checkbox to indicate that the portfolio should be set up as a "Model Portfolio" (see, for example, Figs. 4-6 in our patent application). When this box is checked by the user of the "setupwiz" code, it sets the flag "pSetup->pAds->fModel" to "true" at line 2639 of the "setupwiz" code (page 48).

11. By checking the "Model Portfolio" box, the user of this system then must select one of three options for setting up the account: (a) 100 shares of each stock; (b) $10,000 invested in each security; or (c) dividing a provided investment amount evenly across the set of stocks identified. These options are evaluated in the "setupwiz" code between lines 2679 and 2712 (pages 49-50).

12. Another feature of several claims in our patent application relates to "receiving computer input indicating a past date for purchase of the portfolio" (note, for example, claim 1, line 6; claim 11; claim 17, line 6; claim 27; claim 28, lines 4-5; claim 30, line 5; and claim 34, lines 5-6).

13. At the bottom of the model portfolio property page, there is an option to enter a date. Note, for example, boxes 511 and 603 in Figs. 5 and 6, respectively, of our patent application. This date is captured in the "setupwiz" code at "pSetup->pAds->azMODate," which is evaluated on line 2643 (page 49) and then passed to the call to "LookupPrices" on line 2655 (page 49).

14. Another feature of several claims in our patent application relates to "creating the portfolio and calculating its past performance based on the above inputs" (note, for example, claim 1, lines 7-8; claim 17, lines 7-8; claim 28, lines 6-8; claim 30, lines 8-9; claim 34, lines 9-10; claim 38, lines 6-7; and claim 43, lines 6-7). The combination of the above steps and features of the code function to create the portfolio and determine or calculate its past performance.

15. The "setupwiz" and "setupwiz.h" codes existed in the forms shown in Exhibits B and C prior to August 11, 1999. These computer codes were used as part of a beta test of the invention that began prior to August 11, 1999. Using these codes, the invention performed in its intended manner, as described in our patent application and its claims (Exhibit A), and thus was actually reduced to practice prior to August 11, 1999.

DECLARATION IN LIEU OF OATH

16. I further declare that all information stated herein based upon my own knowledge is true and that all information stated herein based on information and belief is believed to be true, and further that the statements made in this Declaration were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of this application or any patent issuing based on this application.

Date: 8/13/2004 By: P. Hansen
Peter Hansen, Inventor

EXHIBIT A

Current Claims – U.S. Patent Appln. No. 09/592,660

Current Claims – U.S. Patent Appln. No. 09/592,660

The following includes the claims in U.S. Patent Appln. No. 09/592,660 in their present form:

1. (Previously Amended) A computer-implemented method for creating a portfolio of issues comprising the steps of:

receiving computer input identifying specific issues to be added to the portfolio;

receiving computer input indicating a selection of one of a plurality of options for creating the portfolio;

receiving computer input indicating a past date for purchase of the portfolio; and

creating the portfolio and calculating the portfolio's past performance, using a computer, based on said selection of one of said plurality of options and the past date.

2. (Previously Amended) The computer-implemented method according to claim 1, wherein said receiving computer input indicating the selection includes receiving computer input indicating a selection of an option to allocate a number of shares for each issue.

3. (Original) The computer-implemented method according to claim 2, wherein the number of shares is constant for each issue.

4. (Original) The computer-implemented method according to claim 2, wherein the number of shares is not constant for each issue.

5. (Original) The computer-implemented method according to claim 2, wherein the number of shares for each issue is based on a weighting factor.

6. (Previously Amended) The computer-implemented method according to claim 1, wherein said receiving computer input indicating the selection includes receiving computer input indicating a selection of an option to allocate an equal amount for the purchase of each issue and wherein said equal amount is selectable by a user.

7. (Previously Amended) The computer-implemented method according to claim 1, wherein said receiving computer input indicating the selection includes receiving computer input indicating a selection of an option to allocate a total amount for the portfolio.

8. (Original) The computer-implemented method according to claim 7, wherein said amount is equally divided between said issues.

9. (Original) The computer-implemented method according to claim 7, wherein said amount is not equally divided between said issues.

10. (Original) The computer-implemented method according to claim 7, wherein said amount is divided between said issues based on a weighting factor.

11. (Currently Amended) The computer-implemented method according to claim 1, wherein a historical purchase price for each specific issue in the portfolio is obtained, by the computer, from a historical database, based on the past date.

12-16. (Previously Canceled).

17. (Previously Amended) A computer-readable medium having a computer-executable program stored thereon for creating a portfolio of issues, said program comprising the steps of:

receiving identification of specific issues to be added to the portfolio;

receiving a selection of one of a plurality of options for creating the portfolio;

receiving input indicating a past date for purchase of the portfolio; and

creating the portfolio and calculating the portfolio's past performance based on said selection of one of said plurality of options and the past date.

18. (Previously Amended) The computer-readable medium having said program according to claim 17, wherein said receiving the selection includes receiving a selection of an option to allocate a number of shares for each issue.

19. (Original) The computer-readable medium having said program according to claim 18, wherein the number of shares is constant for each issue.

20. (Original) The computer-readable medium having said program according to claim 18, wherein the number of shares is not constant for each issue.

21. (Original) The computer-readable medium having said program according to claim 18, wherein the number of shares for each issue is based on a weighting factor.

22. (Previously Amended) The computer-readable medium having said program according to claim 17, wherein said receiving the selection includes receiving a selection of an option to allocate an equal amount for the purchase of each issue and wherein said equal amount is selectable by a user.

23. (Previously Amended) The computer-readable medium having said program according to claim 17, wherein said receiving the selection includes receiving a selection of an option to allocate a total amount for the portfolio.

24. (Previously Amended) The computer-readable medium having said program according to claim 23, wherein said amount is equally divided between said issues.

25. (Previously Amended) The computer-readable medium having said program according to claim 23, wherein said amount is not equally divided between said issues.

26. (Previously Amended) The computer-readable medium having said program according to claim 23, wherein said amount is divided between said issues based on a weighting factor.

27. (Previously Amended) The computer-readable medium having said program according to claim 17, wherein a historical purchase price for each specific issue in the portfolio is obtained from a historical database, based on the past date.

28. (Previously Amended) A system for creating a portfolio of issues comprising:

an input system for receiving a designation of issues, for receiving a designation of an option for creating said portfolio selected from a plurality of options, and for receiving a historical price associated with each of said issues; and

a processor for determining the number of shares of said issues to add to said portfolio and past performance data relating to the portfolio based on said designated option and said historical price.

29. (Previously Canceled).

30. (Previously Amended) A computer-readable medium having a computer-executable program stored thereon for creating a portfolio of issues, said program comprising the steps of:

receiving identification of specific issues to be added to the portfolio;

receiving identification of a past closing date for the issues;

receiving historical prices for the specific issues based on the past closing date;

receiving a selection of a quantity of said specific issues; and

creating the portfolio based on the specific issues, the selected quantity of the specific issues, and the historical prices.

31. (Previously Amended) The computer-readable medium according to claim 30, wherein said storing step stores said portfolio on a client computer.

32. (Previously Amended) The computer-readable medium according to claim 30, wherein said storing step stores said portfolio on a server computer.

33. (Previously Amended) The computer-readable medium according to claim 30, said program further comprising the step of:

receiving other information regarding the specific issues.

34. (Previously Amended) A computer-implemented method for creating a portfolio of issues comprising the steps of:

receiving computer input identifying specific issues to be added to the portfolio;

receiving computer input identifying a past closing date for the issues;

receiving computer input including historical prices for the specific issues based on the past closing date;

receiving computer input indicating a selection of a quantity of said specific issues; and

creating the portfolio, using a computer, based on the specific issues, the selected quantity of the specific issues, and the historical prices.

35. (Previously Amended) The computer-implemented method according to claim 34, wherein said storing step stores said portfolio on a client computer.

36. (Previously Amended) The computer-implemented method according to claim 34, wherein said storing step stores said portfolio on a server computer.

37. (Previously Amended) The computer-implemented method according to claim 34, further comprising the step of:

receiving other information regarding the specific issues.

38. (Previously Added) A computer-implemented method for creating a portfolio of issues comprising the steps of:

receiving computer input identifying specific issues to be added to the portfolio;

receiving computer input indicating a selection of one of a plurality of options for creating the portfolio; and

creating the portfolio, using the computer, based on said specific issues and said selection of one of said plurality of options.

39. (Previously Added) The computer-implemented method according to claim 38, wherein said receiving computer input indicating the selection includes receiving computer input indicating a selection of an option to allocate a number of shares for each issue.

40. (Previously Added) The computer-implemented method according to claim 38, wherein said receiving computer input indicating the selection includes receiving computer input indicating a selection of an option to allocate an equal amount for the purchase of each issue and wherein said equal amount is selectable by a user.

41. (Previously Added) The computer-implemented method according to claim 38, wherein said receiving computer input indicating the selection includes receiving

computer input indicating a selection of an option to allocate a total amount for the portfolio.

42. (Previously Added) The computer-implemented method according to claim 38, further comprising automatically retrieving current issue price information relating to the specific issues, using the computer, wherein said creating the portfolio further is based on said retrieved current issue price information.

43. (Previously Added) A computer-readable medium having a computer-executable program stored thereon for creating a portfolio of issues, said program comprising the steps of:

receiving identification of specific issues to be added to the portfolio;

receiving a selection of one of a plurality of options for creating the portfolio; and

creating the portfolio based on said specific issues and said selection of one of said plurality of options.

44. (Previously Added) The computer-readable medium having said program according to claim 43, wherein said receiving the selection includes receiving a selection of an option to allocate a number of shares for each issue.

45. (Previously Added) The computer-readable medium having said program according to claim 43, wherein said receiving the selection includes receiving a selection of an option to allocate an equal amount for the purchase of each issue and wherein said equal amount is selectable by a user.

46. (Previously Added) The computer-readable medium having said program according to claim 43, wherein said receiving the selection includes receiving a selection of an option to allocate a total amount for the portfolio.

47. (Previously Added) The computer-readable medium having said program according to claim 43, wherein said program further includes the step of retrieving current issue price information relating to the specific issues, wherein said creating the portfolio further is based on said retrieved current issue price information.

48. (Previously Added) A system for creating a portfolio of issues comprising:

an input system for receiving a designation of issues and for receiving a designation of an option for creating said portfolio selected from a plurality of options; and

a processor for automatically retrieving a price associated with each of said issues and for determining the number of shares of said issues to add to said portfolio based on said designated option and said price.

EXHIBIT B

Copy of "setupwiz" Computer Code

```
1:#include "global.h"
2:#include "dbwin.h"
3:#include "resrc1.h"
4:#include "format.h"
5:#include "buydlg.h"
6:#include "import.h"
7:#include "quicken.h"
8:#include "ofxsess.h"
9:#include "ofx.h"
10:#include "ofxutil.h"
11:#include "security.h"
12:#include "acctdlg.h"
13:#include "tickmisc.h"
14:#include "setupwiz.h"
15:#include "internet.h"
16:#include "monthdlg.h"
17:#include "findsym.h"
18:#include "roaming.h"
19:#include "ofxprofile.h"
20:#include "ofxsignup.h"
21:#include "propsht.h"
22:SZTHISFILE
23:
24:NAW_TIP_STUCT rgTips[] = {
25:     {IDS_NAW_TIP_ACCT,              IDS_PREF_NO_NAW_TIP_ACCT,
IDB_RECORD_ACCT_TIP},
26:     {IDS_NAW_TIP_INV,               IDS_PREF_NO_NAW_TIP_INV,
IDB_RECORD_BUY_TIP},
27:     {IDS_NAW_TIP_ROAM,              IDS_PREF_NO_NAW_TIP_ROAM,
IDB_RECORD_ROAM_TIP},
28:     };
29:
30:
31://--------------------------------------------------------------------------
----------
32:// Setup Wizard Dialog Utilities
33://--------------------------------------------------------------------------
----------
34:void SetFirstOFXPropSheet(HWND hwndDlg, SETUP_STRUCT *pSetup)
35:{
36:     TCHAR   szPassword[MAX_PASSWORD];
37:
38:     pSetup->pOCX->GetMasterPassword(szPassword, MAX_PASSWORD);
39:
40:     if (FileHasEncryptedOFXData(pSetup->pOCX)
41:             || (pSetup->pOCX->GetSecurityLevel() == SECURITY_HIGH) ||
lstrlen(szPassword))
42:     {
43:             assert(pSetup->pOFX);
44:             pSetup->pOFX->SetPasswordDlgParent(hwndDlg);
45:
46:             if (pSetup->pOFX->InitOFX(FALSE))
47:                     SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_OFX_1);
48:             else
49:                     SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
50:     }
51:     else
52:             SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_MASTER_PASSWORD);
53:}
54:
55://--------------------------------------------------------------------------
----------
56:// Checks to see if there is data to convert when user clicks next
```

```
57://-------------------------------------------------------------------------
---------
58:BOOL CALLBACK AccountWizFTUEDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
59:{
60:     SETUP_STRUCT    *pSetup = NULL;
61:
62:     switch(uMsg)
63:     {
64:     case WM_PAINT:
65:             PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
66:             break;
67:
68:     case WM_NOTIFY:
69:             pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
70:             switch (((NMHDR *)lParam)->code)
71:             {
72:             case PSN_SETACTIVE:
73:                     PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_NEXT);
74:                     break;
75:
76:             case PSN_WIZNEXT:
77:                     // Setup the sample portfolios.
78:                     return TRUE;
79:
80:             }
81:             break;
82:
83:     case WM_INITDIALOG:
84:             pSetup = (SETUP_STRUCT *)((INVPROPSHEETPAGE *)lParam)->lParam;
85:             SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
86:             return TRUE;
87:     }
88:
89:     return(FALSE);
90:}
91:
92:
93://-------------------------------------------------------------------------
---------
94:// Dialog proc for the setup wizard New Account Property Sheet.
95://-------------------------------------------------------------------------
---------
96:BOOL CALLBACK AccountWizNewDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
97:{
98:     SETUP_STRUCT    *pSetup = NULL;
99:     int watchFlag = 0;
100:
101:    switch(uMsg)
102:    {
103:    case WM_PAINT:
104:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
105:            break;
106:
107:    case WM_COMMAND:
108:            switch(HIWORD(wParam))
109:            {
110:            case BN_DBLCLK:
111:                    PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
112:                    break;
```

```
113:            }
114:
115:            switch(LOWORD(wParam))
116:            {
117:            case IDC_ACCOUNT_REGULAR:
118:                    SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_REGULAR));
119:                    break;
120:
121:            case IDC_ACCOUNT_WATCH:
122:                    SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_WATCH));
123:                    break;
124:
125:            case IDC_ACCOUNT_IMPORT_EXISTING:
126:                    SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_IMPORT));
127:                    break;
128:            }
129:            break;
130:
131:    case WM_NOTIFY:
132:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
133:            assert(pSetup);
134:            switch (((NMHDR *)lParam)->code)
135:            {
136:            case PSN_SETACTIVE:
137:                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_NEXT);
138:                    break;
139:
140:            case PSN_WIZNEXT:
141:                    SW_ACCT_TYPE            swatSelect;
142:
143:                    swatSelect = SWAT_NONE;
144:
145:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_REGULAR))
146:                            swatSelect = SWAT_REGULAR;
147:
148:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_WATCH))
149:                            swatSelect = SWAT_WATCH;
150:
151:                    if (IsDlgButtonChecked(hwndDlg,
IDC_ACCOUNT_IMPORT_EXISTING))
152:                            swatSelect = SWAT_IMPORT;
153:
154:                    switch (swatSelect)
155:                    {
156:                    case SWAT_REGULAR:
157:                            if (pSetup->pAds == NULL)
158:                            {
159:                                    pSetup->pAds = new ACCTDLG_STRUCT;
160:                                    assert(pSetup->pAds);
161:                                    my_memset(pSetup->pAds, 0,
sizeof(ACCTDLG_STRUCT));
162:                            }
163:
164:                            pSetup->pAds->fWatch = FALSE;
165:
166:                            SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_REGULAR);
167:                            break;
168:
169:                    case SWAT_WATCH:
```

```
                                    setupwizlines
170:                                if (pSetup->pAds == NULL)
171:                                {
172:                                        pSetup->pAds = new ACCTDLG_STRUCT;
173:                                        assert(pSetup->pAds);
174:                                        my_memset(pSetup->pAds, 0,
sizeof(ACCTDLG_STRUCT));
175:                                }
176:
177:                                pSetup->pAds->fWatch = TRUE;
178:
179:                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WATCH);
180:                                break;
181:
182:                        case SWAT_IMPORT:
183:                                SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
184:                                break;
185:
186:                        default:
187:                                assert(FALSE);
188:                                SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
189:                                break;
190:                        }
191:
192:                        pSetup->swatSelect = swatSelect;
193:
194:                        return TRUE;
195:                }
196:                break;
197:
198:        case WM_INITDIALOG:
199:                pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
200:                SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
201:
202:                CheckDlgButton(hwndDlg, IDC_ACCOUNT_REGULAR, BST_CHECKED);
203:
204:                SetDlgItemText(hwndDlg, IDC_ACCOUNT_DESCRIPTION,
GetRscString(IDS_ACCT_DESC_REGULAR));
205:
206:                if (GET_ROAMING(pSetup->pOCX)->IsSavingData())
207:                        EnableControl(hwndDlg, IDC_ACCOUNT_IMPORT_EXISTING, FALSE);
208:
209:                return(TRUE);
210:        }
211:
212:        return(FALSE);
213:}
214:
215://-------------------------------------------------------------------------
-----------
216:// Dialog proc for the setup wizard Import Existing Account Property Sheet.
217://-------------------------------------------------------------------------
-----------
218:BOOL CALLBACK AccountWizImportExistingDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
219:{
220:        SETUP_STRUCT    *pSetup = NULL;
221:        RECT                    rcTop, rcBottom, rcSquare, rcQuickenDC, rcOFX;
222:        int                     iOffset = 0;
223:
224:        switch(uMsg)
225:        {
```

```
226:    case WM_PAINT:
227:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
228:            break;
229:
230:    case WM_COMMAND:
231:            switch(HIWORD(wParam))
232:            {
233:            case BN_DBLCLK:
234:                    PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
235:                    break;
236:            }
237:            break;
238:
239:    case WM_NOTIFY:
240:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
241:            assert(pSetup);
242:            switch (((NMHDR *)lParam)->code)
243:            {
244:            case PSN_SETACTIVE:
245:                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
246:                    break;
247:
248:            case PSN_WIZBACK:
249:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
250:                    return TRUE;
251:
252:            case PSN_WIZNEXT:
253:                    SW_ACCT_TYPE            swatSelect;
254:
255:                    swatSelect = SWAT_IMPORT;
256:
257:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY))
258:                            swatSelect = SWAT_MONEY;
259:
260:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_QUICKEN))
261:                            swatSelect = SWAT_QUICKEN;
262:
263:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_YAHOO))
264:                            swatSelect = SWAT_YAHOO;
265:
266:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM))
267:                            swatSelect = SWAT_QUICKEN_COM;
268:
269:                    if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_OFX))
270:                            swatSelect = SWAT_OFX;
271:
272:                    switch (swatSelect)
273:                    {
274:                    case SWAT_MONEY:
275:                            if (pSetup->pMoney == NULL){
276:                                    pSetup->pMoney = new CMoney(pSetup->pOCX,
GET_DATA(pSetup->pOCX));
277:                                    assert(pSetup->pMoney);
278:                                    pSetup->pMoney->AddRef();
279:
pSetup->pMoney->SetHWndParent(GetParent(hwndDlg));
280:                            }
281:                            SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_1);
282:                            break;
283:
```

```
284:                    case SWAT_QUICKEN:
285:                            if (pSetup->pQuicken == NULL){
286:                                    pSetup->pQuicken = new
CQuicken(pSetup->pOCX, GET_DATA(pSetup->pOCX));
287:                                    assert(pSetup->pQuicken);
288:                                    pSetup->pQuicken->AddRef();
289:
pSetup->pQuicken->SetHWndParent(GetParent(hwndDlg));
290:                            }
291:                            SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_QUICKEN);
292:                            break;
293:
294:                    case SWAT_YAHOO:
295:                    case SWAT_QUICKEN_COM:
296:                            if (pSetup->pWeb == NULL){
297:                                            pSetup->pWeb = new
CWeb(pSetup->pOCX, GET_DATA(pSetup->pOCX));
298:                                            assert(pSetup->pWeb);
299:                                            pSetup->pWeb->AddRef();
300:
pSetup->pWeb->SetHWndParent(GetParent(hwndDlg));
301:                            }
302:
303:                            if (swatSelect == SWAT_YAHOO)
304:                                    pSetup->pWeb->setOC(YAHOO);
305:                            else if (swatSelect == SWAT_QUICKEN_COM)
306:                                    pSetup->pWeb->setOC(QUICKENDC);
307:                            else
308:                                    assert(FALSE);
309:
310:                            SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WEB);
311:                            break;
312:
313:                    case SWAT_OFX:
314:                            if (pSetup->pOFX == NULL)
315:                            {
316:                                    pSetup->pOFX = new COFX(pSetup->pOCX,
GET_DATA(pSetup->pOCX));
317:                                    assert(pSetup->pOFX);
318:                                    pSetup->pOFX->AddRef();
319:
pSetup->pOFX->SetHWndParent(GetParent(hwndDlg));
320:                            }
321:
322:                            SetFirstOFXPropSheet(hwndDlg, pSetup);
323:                            break;
324:
325:                    default:
326:                            assert(FALSE);
327:                            SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
328:                            break;
329:                    }
330:
331:                    pSetup->swatSelect = swatSelect;
332:                    return TRUE;
333:            }
334:            break;
335:
336:    case WM_INITDIALOG:
337:            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
338:            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
```

```
339:            //should we resize?
340:            if (pSetup->pOCX->IsYahooEnabled() == FALSE ||
341:                    pSetup->pOCX->IsQuickenDotComEnabled() == FALSE) {
342:                    if ((pSetup->pOCX->IsYahooEnabled() == FALSE) &&
343:                            (pSetup->pOCX->IsQuickenDotComEnabled() == FALSE)) {
344:                            HideControl(hwndDlg, IDC_ACCOUNT_YAHOO);
345:                            HideControl(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM);
346:
347:                            GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcBottom);
348:                            ScreenToClient(hwndDlg, (POINT *)&rcBottom);
349:
350:                            GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKEN,
&rcTop);
351:                            ScreenToClient(hwndDlg, (POINT *)&rcTop);
352:                    } else if (pSetup->pOCX->IsYahooEnabled() == FALSE) {
353:                            HideControl(hwndDlg, IDC_ACCOUNT_YAHOO);
354:
355:                            GetControlRect(hwndDlg, IDC_ACCOUNT_YAHOO,
&rcBottom);
356:                            ScreenToClient(hwndDlg, (POINT *)&rcBottom);
357:
358:                            GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKEN,
&rcTop);
359:                            ScreenToClient(hwndDlg, (POINT *)&rcTop);
360:
361:                            GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcQuickenDC);
362:                            ScreenToClient(hwndDlg, (POINT *)&rcQuickenDC);
363:                    } else if (pSetup->pOCX->IsQuickenDotComEnabled() == FALSE)
{
364:                            HideControl(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM);
365:
366:                            GetControlRect(hwndDlg, IDC_ACCOUNT_QUICKENDOTCOM,
&rcBottom);
367:                            ScreenToClient(hwndDlg, (POINT *)&rcBottom);
368:
369:                            GetControlRect(hwndDlg, IDC_ACCOUNT_YAHOO, &rcTop);
370:                            ScreenToClient(hwndDlg, (POINT *)&rcTop);
371:                    }
372:                    // Find out how far to move the items.
373:                    iOffset = rcBottom.bottom - rcTop.bottom;
374:
375:                    GetControlRect(hwndDlg, IDC_IMPORT_SQUARE, &rcSquare);
376:
377:                    if (pSetup->pOCX->IsQuickenDotComEnabled() != FALSE) {
378:                            // Move QuickenDotCom up if needed
379:                            SetWindowPos(GetDlgItem(hwndDlg,
IDC_ACCOUNT_QUICKENDOTCOM), NULL,
380:                                                    rcQuickenDC.left, rcQuickenDC.top -
iOffset, 0, 0,
381:                                                    SWP_NOZORDER | SWP_NOSIZE |
SWP_NOACTIVATE);
382:                    }
383:
384:                    // Move OFX up if needed
385:                    GetControlRect(hwndDlg, IDC_ACCOUNT_OFX, &rcOFX);
386:                    ScreenToClient(hwndDlg, (POINT *)&rcOFX);
387:                    SetWindowPos(GetDlgItem(hwndDlg, IDC_ACCOUNT_OFX), NULL,
388:                                    rcOFX.left, rcOFX.top - iOffset, 0, 0,
389:                                    SWP_NOZORDER | SWP_NOSIZE |
SWP_NOACTIVATE);
390:
```

```
391:                    // Resize the frame
392:                    SetWindowPos(GetDlgItem(hwndDlg, IDC_IMPORT_SQUARE), NULL,
0, 0,
393:                            rcSquare.right - rcSquare.left, rcSquare.bottom -
rcSquare.top - iOffset,
394:                            SWP_NOZORDER | SWP_NOMOVE | SWP_NOACTIVATE);
395:
396:            }
397:            CheckDlgButton(hwndDlg, IDC_ACCOUNT_MONEY, BST_CHECKED);
398:
399:            return(TRUE);
400:    }
401:
402:    return(FALSE);
403:}
404:
405://-------------------------------------------------------------------------
-----------
406:// Dialog proc for the Account Wizard Regular Property Sheet.
407://-------------------------------------------------------------------------
-----------
408:BOOL CALLBACK DoesEditBoxHaveText(HWND hwndDlg, long idcItem)
409:{
410:    TCHAR   szText[MAX_STRING];
411:    BOOL    fResult = FALSE;
412:
413:    szText[0] = 0;
414:    if (GetDlgItemText(hwndDlg, idcItem, szText, MAX_STRING))
415:            alltrim(szText);
416:
417:    if (strlen(szText) > 0)
418:            fResult = TRUE;
419:
420:    return(fResult);
421:}
422:
423://-------------------------------------------------------------------------
-----------
424:// Dialog proc for the Account Wizard Regular Property Sheet.
425://-------------------------------------------------------------------------
-----------
426:BOOL CALLBACK AccountWizRegularDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
427:{
428:    SETUP_STRUCT    *pSetup = NULL;
429:
430:    switch(uMsg)
431:    {
432:    case WM_USER_SETFOCUS:
433:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
434:            if (pSetup->pAds->fErrorDirty)
435:                    DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
436:            pSetup->pAds->fErrorDirty = FALSE;
437:            break;
438:
439:    case WM_COMMAND:
440:            switch(HIWORD(wParam))
441:            {
442:            case EN_CHANGE:
443:                    if (DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME))
444:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
445:                    else
```

```
446:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
447:                    break;
448:            }
449:
450:            switch(LOWORD(wParam))
451:            {
452:            case IDC_NEWACCT_CASH:
453:                    goto lbEnableCashBalance;
454:                    break;
455:            }
456:            break;
457:
458:    case WM_PAINT:
459:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
460:            break;
461:
462:    case WM_NOTIFY:
463:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
464:            assert(pSetup);
465:            switch (((NMHDR *)lParam)->code)
466:            {
467:            case PSN_SETACTIVE:
468:                    if (DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME))
469:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
470:                    else
471:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
472:
473:lbEnableCashBalance:
474:                    if (IsDlgButtonChecked(hwndDlg, IDC_NEWACCT_CASH))
475:                    {
476:                            EnableControl(hwndDlg, IDC_NEWACCT_CASHBAL, TRUE);
477:                            EnableHotKey(hwndDlg, IDC_NEWACCT_CASHBAL_TEXT, 'I',
TRUE);
478:
479:                    }
480:                    else
481:                    {
482:                            EnableControl(hwndDlg, IDC_NEWACCT_CASHBAL, FALSE);
483:                            EnableHotKey(hwndDlg, IDC_NEWACCT_CASHBAL_TEXT, 'I',
FALSE);
484:
485:                    }
486:                    break;
487:
488:            case PSN_WIZFINISH:
489:                    break;
490:
491:            case PSN_WIZBACK:
492:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
493:                    return TRUE;
494:
495:            case PSN_WIZNEXT:
496:                    int                     resIdError = IDS_NEED_ACCOUNT_NAME;
497:                    int                     iBadControl =
IDC_NEWACCT_ACCOUNT_NAME;
498:                    BOOL            fSuccess = FALSE;
499:                    double          dbl = 0;
500:
501:                    // Validate the Account Name.
```

```
502:                    if (GetDlgItemText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME,
pSetup->pAds->szAccountName, MAX_ACCOUNT_NAME))
503:                    {
504:                            // Make sure we have a name
505:                            alltrim(pSetup->pAds->szAccountName);
506:                            fSuccess = lstrlen(pSetup->pAds->szAccountName) > 0;
507:                    }
508:
509:                    // Check if account already exists
510:                    if (fSuccess)
511:                    {
512:                            resIdError = IDS_ACCOUNT_NAME_IN_USE;
513:                            fSuccess =
GET_DATA(pSetup->pOCX)->FindAccountName(pSetup->pAds->szAccountName) == NULL;
514:                    }
515:
516:                    // Validatation Cash balance for basic format -- numerals,
commas, etc.
517:                    if (fSuccess)
518:                    {
519:                            iBadControl = IDC_NEWACCT_CASHBAL;
520:
521:                            fSuccess = DlgValidateQuantity(hwndDlg,
IDC_NEWACCT_CASHBAL, &resIdError);
522:
523:                            // get the purchase price
524:                            if (fSuccess)
525:                                    fSuccess = DlgGetDouble(hwndDlg,
IDC_NEWACCT_CASHBAL, BDGD_MAX_LONG, &dbl, &resIdError);
526:                    }
527:
528:                    // If we're still ok, go to the finish screen, otherwise
error
529:                    if (fSuccess)
530:                    {
531:                            pSetup->pAds->dblCash = dbl;
532:                            pSetup->pAds->fCash      =
IsDlgButtonChecked(hwndDlg, IDC_NEWACCT_CASH);
533:                            PropSheet_PressButton(GetParent(hwndDlg),
PSBTN_FINISH);
534:                    }
535:                    else
536:                    {
537:                            pSetup->pAds->fErrorDirty = TRUE;
538:                            my_beep();
539:                            PostMessage(hwndDlg, WM_USER_SETFOCUS,
(WPARAM)iBadControl, resIdError);
540:                    }
541:
542:                    // Don't let the next button take us anywhere...
543:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
544:                    return(TRUE);
545:            }
546:            break;
547:
548:
549:    case WM_INITDIALOG:
550:            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
551:            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
552:
553:            SubClassNumberField(hwndDlg, IDC_NEWACCT_CASHBAL, FALSE);
554:
555:            // Set label on Finish button to Next
```

```
556:
557:            return(TRUE);
558:    }
559:
560:    return(FALSE);
561:}
562:
563:
564://-------------------------------------------------------------------------
-----------
565:// Dialog proc for the Account Wizard Regular Property Sheet.
566://-------------------------------------------------------------------------
-----------
567:void CheckWatchAccountButtonStatus(HWND hwndDlg)
568:{
569:    if ((DoesEditBoxHaveText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME)) &&
570:            (DoesEditBoxHaveText(hwndDlg, IDC_WATCH_SYMBOLS)))
571:            PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_FINISH);
572:    else
573:            PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_DISABLEDFINISH);
574:}
575:
576://-------------------------------------------------------------------------
-----------
577:// Dialog proc for the Account Wizard Regular Property Sheet.
578://-------------------------------------------------------------------------
-----------
579:BOOL CALLBACK AccountWizWatchDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
580:{
581:    SETUP_STRUCT    *pSetup = NULL;
582:
583:    switch(uMsg)
584:    {
585:    case WM_USER_SETFOCUS:
586:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
587:            if (pSetup->pAds->fErrorDirty)
588:                    DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
589:            pSetup->pAds->fErrorDirty = FALSE;
590:            break;
591:
592:    case WM_COMMAND:
593:            switch(HIWORD(wParam))
594:            {
595:            case EN_CHANGE:
596:                    CheckWatchAccountButtonStatus(hwndDlg);
597:                    break;
598:            }
599:
600:            switch(LOWORD(wParam))
601:            {
602:            case IDCANCEL:
603:                    // Wierd: v7 bug 2186: Multiline edit control does not pass
604:                    //  the cancel method up to the parent, so we have to catch
it here
605:                    //  and force it ourselves.
606:                    PropSheet_PressButton(GetParent(hwndDlg), PSBTN_CANCEL);
607:                    break;
608:
609:            case IDC_WATCH_FIND_SYMBOL:
610:                    pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
```

```
SETUPPROP_PROP_NAME);
611:
612:                    // Call up the find symbol dialog.  It returns TRUE if the
user pressed
613:                    // OK to exit it. szTicker is the ticker selected by the
user in the Find
614:                    // Symbol dialog.
615:                    {
616:                            CLISTATUS status;
617:                            OCX(pSetup->pOCX)->InitWosaStatusStruct(&status,
qrsPortFindSymbol);
618:
619:                            TCHAR   szSymbols[MAX_STRING * 2] = {0};
620:                            TCHAR   szTicker[MAX_STRING] = {0};
621:                            int             cch;
622:
623:                            if (FindSymbolDlg(hwndDlg, szTicker,
OCX(pSetup->pOCX)->GetDefaultTickerCurid(),
624:                                    ((CPortfolioControl
*)pSetup->pOCX)->m_spClientSite, &status, TRUE,
625:                                    FALSE, INV_STOCK)
626:                                    && lstrlen(szTicker))
627:                            {
628:                                    SendDlgItemMessage(hwndDlg,
IDC_WATCH_SYMBOLS, WM_GETTEXT, MAX_STRING, (LPARAM)szSymbols);
629:                                    alltrim(szSymbols);
630:
631:                                    if (lstrlen(szSymbols) > 0)
632:                                            StrCatMaxLen(szSymbols, ", ",
sizeof(szSymbols));
633:
634:                                    StrCatMaxLen(szSymbols, szTicker,
sizeof(szSymbols));
635:
636:                                    // Chop off the szSymbols to fit into 256
characters..
637:                                    if (lstrlen(szSymbols) > 256)
638:                                    {
639:                                            cch = 255;
640:
641:                                            for (cch = 255; (cch > 0) &&
(szSymbols[cch] != ','); cch--)
642:                                                    szSymbols[cch] = '\0';
643:                                    }
644:
645:                                    SendDlgItemMessage(hwndDlg,
IDC_WATCH_SYMBOLS, WM_SETTEXT, MAX_STRING, (LPARAM)szSymbols);
646:                                    CheckWatchAccountButtonStatus(hwndDlg);
647:                            }
648:                    }
649:                    break;
650:
651:            case IDC_WATCH_ADVANCED:
652:                    if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_ADVANCED))
653:                    {
654:                            ShowWindow(GetDlgItem(hwndDlg, IDC_MODEL_OPTIONS),
SW_SHOW);
655:                            ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DOLLARS),
SW_SHOW);
656:                            ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_SHARES),
SW_SHOW);
657:                            ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_TOTAL),
SW_SHOW);
```

```
658:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_TOTAL_EDIT), SW_SHOW);
659:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_DATE_CAPTION), SW_SHOW);
660:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DATE),
SW_SHOW);
661:                                if (!g_fOldComCtl)
662:                                        ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_CALICON), SW_SHOW);
663:
664:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_ADVANCED_DESCRIPTION), SW_HIDE);
665:                        }
666:                        else
667:                        {
668:                                ShowWindow(GetDlgItem(hwndDlg, IDC_MODEL_OPTIONS),
SW_HIDE);
669:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DOLLARS),
SW_HIDE);
670:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_SHARES),
SW_HIDE);
671:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_TOTAL),
SW_HIDE);
672:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_TOTAL_EDIT), SW_HIDE);
673:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_WATCH_DATE_CAPTION), SW_HIDE);
674:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_DATE),
SW_HIDE);
675:                                ShowWindow(GetDlgItem(hwndDlg, IDC_WATCH_CALICON),
SW_HIDE);
676:
677:                                ShowWindow(GetDlgItem(hwndDlg,
IDC_ADVANCED_DESCRIPTION), SW_SHOW);
678:                        }
679:                        break;
680:
681:                case IDC_WATCH_DOLLARS:
682:                case IDC_WATCH_SHARES:
683:                case IDC_WATCH_TOTAL:
684:                        if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_TOTAL))
685:                        {
686:                                EnableControl(hwndDlg, IDC_WATCH_TOTAL_EDIT, TRUE);
687:                        }
688:                        else
689:                        {
690:                                EnableControl(hwndDlg, IDC_WATCH_TOTAL_EDIT, FALSE);
691:                        }
692:                        break;
693:
694:                case IDC_WATCH_CALICON:
695:                        DoMonthDlg(hwndDlg, IDC_WATCH_DATE, IDC_WATCH_CALICON);
696:                        break;
697:                }
698:                break;
699:
700:        case WM_PAINT:
701:                PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
702:                break;
703:
704:        case WM_NOTIFY:
705:                pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
```

```
706:            assert(pSetup);
707:            switch (((NMHDR *)lParam)->code)
708:            {
709:            case PSN_SETACTIVE:
710:                    CheckWatchAccountButtonStatus(hwndDlg);
711:                    break;
712:
713:            case PSN_WIZBACK:
714:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_NEW);
715:                    return TRUE;
716:
717:            case PSN_WIZFINISH:
718:                    int                     resIdError = IDS_NEED_ACCOUNT_NAME;
719:                    int                     iBadControl =
IDC_NEWACCT_ACCOUNT_NAME;
720:                    BOOL            fSuccess = FALSE;
721:                    double          dbl = 0;
722:
723:                    // Validate the Account Name.
724:                    if (GetDlgItemText(hwndDlg, IDC_NEWACCT_ACCOUNT_NAME,
pSetup->pAds->szAccountName, MAX_ACCOUNT_NAME))
725:                    {
726:                            // Make sure we have a name
727:                            alltrim(pSetup->pAds->szAccountName);
728:                            fSuccess = lstrlen(pSetup->pAds->szAccountName) > 0;
729:                    }
730:
731:                    // Check if account already exists
732:                    if (fSuccess)
733:                    {
734:                            resIdError = IDS_ACCOUNT_NAME_IN_USE;
735:                            fSuccess =
GET_DATA(pSetup->pOCX)->FindAccountName(pSetup->pAds->szAccountName) == NULL;
736:                    }
737:
738:                    if (fSuccess)
739:                    {
740:                            resIdError = IDS_ACCOUNT_NEED_SYMBOLS;
741:                            iBadControl = IDC_WATCH_SYMBOLS;
742:
743:                            // Get list of symbols
744:                            if (GetDlgItemText(hwndDlg, IDC_WATCH_SYMBOLS,
pSetup->pAds->szSymbols, MAX_STRING))
745:                            {
746:                                    // Make sure we have a name
747:                                    alltrim(pSetup->pAds->szSymbols);
748:                                    fSuccess = lstrlen(pSetup->pAds->szSymbols)
> 0;
749:                            }
750:                    }
751:
752:                    // Get advanced options if visible.
753:                    if (fSuccess)
754:                    {
755:                            if (IsDlgButtonChecked(hwndDlg, IDC_WATCH_ADVANCED))
756:                            {
757:                                    pSetup->pAds->fModel = TRUE;
758:
759:                                    if (IsDlgButtonChecked(hwndDlg,
IDC_WATCH_DOLLARS))
760:                                            pSetup->pAds->mo = MO_DOLLARS;
761:
762:                                    if (IsDlgButtonChecked(hwndDlg,
```

```
IDC_WATCH_SHARES))
763:                                                    pSetup->pAds->mo = MO_SHARES;
764:
765:                                            if (IsDlgButtonChecked(hwndDlg,
IDC_WATCH_TOTAL))
766:                                            {
767:                                                    TCHAR   szTotal[MAX_STRING];
768:
769:                                                    iBadControl = IDC_WATCH_TOTAL_EDIT;
770:                                                    resIdError = IDS_NEED_POSITIVE;
771:                                                    fSuccess = FALSE;
772:
773:                                                    pSetup->pAds->mo = MO_TOTAL;
774:                                                    if (GetDlgItemText(hwndDlg,
IDC_WATCH_TOTAL_EDIT, szTotal, MAX_STRING))
775:                                                    {
776:                                                            // Make sure we have a name
777:                                                            alltrim(szTotal);
778:                                                            pSetup->pAds->dblMOTotal =
my_atol(szTotal);
779:
780:                                                            fSuccess =
pSetup->pAds->dblMOTotal > 0;
781:                                                    }
782:                                            }
783:
784:                                            if (fSuccess)
785:                                            {
786:                                                    CDate                   date, today;
787:                                                    SYSTEMTIME              st;
788:
789:                                                    GetLocalTime(&st);
790:                                                    today.SetDate(&st);
791:
792:                                                    resIdError = IDS_NEED_MODEL_DATE;
793:                                                    iBadControl = IDC_WATCH_DATE;
794:
795:                                                    GetDlgItemText(hwndDlg,
IDC_WATCH_DATE, pSetup->pAds->szMODate, MAX_STRING);
796:
797:                                                    if (!SetDateToString(&date,
pSetup->pAds->szMODate, NULL))
798:                                                            fSuccess = FALSE;
799:                                                    else if (date > today)
800:                                                            fSuccess = FALSE;
801:                                            }
802:                                    }
803:                            }
804:
805:                            // If we're not ok, error
806:                            if (!fSuccess)
807:                            {
808:                                    pSetup->pAds->fErrorDirty = TRUE;
809:                                    my_beep();
810:                                    PostMessage(hwndDlg, WM_USER_SETFOCUS,
(WPARAM)iBadControl, resIdError);
811:                                    SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
812:                            }
813:                            return(TRUE);
814:                    }
815:                    break;
816:
817:    case WM_INITDIALOG:
```

```
818:            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
819:            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
820:
821:            // Set the sample tickers
822:            // SetDlgItemText(hwndDlg, IDC_WATCH_SYMBOLS, "MSFT, INTC, DELL");
823:
824:            // Set the default option
825:            CheckDlgButton(hwndDlg, IDC_WATCH_SHARES, BST_CHECKED);
826:
827:            // Subclass number fields
828:            SubClassNumberField(hwndDlg, IDC_WATCH_TOTAL_EDIT, FALSE);
829:
830:            // Set the total portolio value to $10 000
831:            SetDlgItemText(hwndDlg, IDC_WATCH_TOTAL_EDIT, "10000");
832:
833:            // Set up the calendar bitmap button
834:            SetCalendarButton(hwndDlg, IDC_WATCH_CALICON, IDB_CALICON);
835:
836:            // Subclass the date field to enable +/- keys
837:            SubClassDateEditField(hwndDlg, IDC_WATCH_DATE);
838:
839:            // Set the date initially to today.
840:            {
841:                    SYSTEMTIME      st;
842:                    TCHAR           szTemp[MAX_STRING];
843:                    GetLocalTime(&st);
844:                    CDate           today;
845:
846:                    today.SetDate(&st);
847:
848:                    SetDlgItemText(hwndDlg, IDC_WATCH_DATE,
today.Format(szTemp));
849:            }
850:
851:            // Check for enabling/disabling of advanced checkbox:
IDC_WATCH_ADVANCED
852:
853:            return(TRUE);
854:
855:    case WM_DESTROY:
856:            FreeCalendarBitmap(hwndDlg);
857:            return(0);
858:
859:    }
860:
861:    return(FALSE);
862:}
863:
864://---------------------------------------------------------------------------
-----------
865:// Dialog proc for the Acocunt Wizard Finish Property Sheet. (Shared by all
account types)
866://---------------------------------------------------------------------------
-----------
867:BOOL CALLBACK AccountWizFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
868:{
869:    SETUP_STRUCT    *pSetup = NULL;
870:
871:    switch(uMsg)
872:    {
873:    case WM_PAINT:
874:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
```

```
nDlgBmpWidth, nDlgBmpHeight, TRUE);
875:            break;
876:
877:    case WM_NOTIFY:
878:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
879:            assert(pSetup);
880:            switch (((NMHDR *)lParam)->code)
881:            {
882:            case PSN_SETACTIVE:
883:                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_FINISH);
884:                    break;
885:
886:            case PSN_WIZBACK:
887:                    // We don't know where we are going - the calling dialog
proc should handle this message.
888:                    assert(FALSE);
889:                    break;
890:
891:            }
892:            break;
893:
894:    case WM_INITDIALOG:
895:            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
896:            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
897:            return(TRUE);
898:    }
899:
900:    return(FALSE);
901:}
902:
903://-------------------------------------------------------------------------
----------
904:// Dialog proc for the Account Wizard Web Password Property Sheet.
905://-------------------------------------------------------------------------
----------
906:BOOL CALLBACK AccountWizWebDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
907:{
908:    SETUP_STRUCT    *pSetup = NULL;
909:    TCHAR                   tmpID[MAX_STRING];
910:    TCHAR                   tmpPassword[MAX_STRING];
911:
912:    switch(uMsg)
913:    {
914:    case WM_PAINT:
915:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
916:            break;
917:
918:    case WM_USER_WEBTXT_DOWNLOAD:
919:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
920:            assert(pSetup->pWeb);
921:
922:            if ((DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_WEB_USERID)) &&
923:                    (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_WEB_PASSWORD)))
924:            {
925:                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
926:            }
927:            else
928:            {
929:                    PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
```

```
930:            }
931:            break;
932:
933:    case WM_COMMAND:
934:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
935:            assert(pSetup);
936:            switch(HIWORD(wParam))
937:            {
938:            case EN_CHANGE:
939:                    if ((pSetup->pWeb->isDownloaded() != NOTHING) &&
940:                            (DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_USERID)) &&
941:                            (DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_PASSWORD)))
942:                    {
943:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
944:                    }
945:                    else
946:                    {
947:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
948:                    }
949:                    break;
950:            }
951:            break;
952:
953:    case WM_NOTIFY:
954:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
955:            assert(pSetup);
956:            switch (((NMHDR *)lParam)->code)
957:            {
958:            case PSN_SETACTIVE:
959:                    assert(pSetup->pWeb);
960:
961:                    pSetup->pWeb->SetWebPrompts(hwndDlg);
962:
963:                    if ((pSetup->pWeb->EnsureWebTxt(NULL, hwndDlg))
964:                            || (!DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_USERID))
965:                            || (!DoesEditBoxHaveText(hwndDlg,
IDC_ACCOUNT_WEB_PASSWORD)))
966:                    {
967:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
968:                    }
969:                    else
970:                    {
971:                            PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
972:                    }
973:                    break;
974:
975:            case PSN_WIZBACK:
976:                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
977:                    return TRUE;
978:
979:            case PSN_WIZNEXT:
980:                    //Get user name and password
981:                    GetDlgItemText(hwndDlg, IDC_ACCOUNT_WEB_USERID, tmpID,
MAX_STRING);
982:                    GetDlgItemText(hwndDlg, IDC_ACCOUNT_WEB_PASSWORD,
```

```
tmpPassword, MAX_STRING);
983:                    alltrim(tmpID);
984:                    alltrim(tmpPassword);
985:                    if (my_strcmp(tmpID, pSetup->pWeb->checkID())
986:                            || my_strcmp(tmpPassword,
pSetup->pWeb->checkPassword())) {
987:                            pSetup->pWeb->ClearAccounts();
988:                    }
989:                    pSetup->pWeb->setUserID(tmpID);
990:                    pSetup->pWeb->setPassword(tmpPassword);
991:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB_LIST);
992:                    return(TRUE);
993:
994:            }
995:            break;
996:
997:    case WM_INITDIALOG:
998:            pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
999:            SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1000:
1001:           assert(pSetup);
1002:
1003:           return(TRUE);
1004:   }
1005:
1006:   return(FALSE);
1007:}
1008:
1009://-------------------------------------------------------------------------
-----------
1010:// Dialog proc for the Account Wizard Money 2 Property Sheet.
1011://-------------------------------------------------------------------------
-----------
1012:void CheckImportWizButtonStatus(SETUP_STRUCT *pSetup, HWND hwndDlg)
1013:{
1014:   // Check the selection
1015:   if (SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST, LB_GETSELCOUNT, 0, 0) > 0)
1016:   {
1017:           if (pSetup->fNext == FALSE)
1018:           {
1019:                   PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
1020:                   pSetup->fNext = TRUE;
1021:           }
1022:   }
1023:   else
1024:   {
1025:           if (pSetup->fNext == TRUE)
1026:           {
1027:                   PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
1028:                   pSetup->fNext = FALSE;
1029:           }
1030:   }
1031:}
1032:
1033://-------------------------------------------------------------------------
-----------
1034:// Dialog proc for the Account Wizard Web Property Sheet.
1035://-------------------------------------------------------------------------
-----------
1036:BOOL CALLBACK AccountWizWebListDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1037:{
```

```
1038:   SETUP_STRUCT    *pSetup = NULL;
1039:
1040:   switch(uMsg)
1041:   {
1042:   case WM_PAINT:
1043:           PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1044:           break;
1045:
1046:   case WM_USER_PORT_LIST_DOWNLOAD:
1047:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1048:           assert(pSetup->pWeb);
1049:
1050:           // Populate the list with the accounts.
1051:           pSetup->pWeb->FillAccountList(hwndDlg, NULL);
1052:
1053:           // Update the navigation elements.
1054:           CheckImportWizButtonStatus(pSetup, hwndDlg);
1055:
1056:           break;
1057:
1058:   case WM_USER_BAD_LOGIN:
1059:           PropSheet_PressButton(GetParent(hwndDlg), PSBTN_BACK);
1060:                   break;
1061:
1062:   case WM_COMMAND:
1063:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1064:           switch(LOWORD(wParam))
1065:           {
1066:           case IDC_ACCOUNT_LIST:
1067:                   if (HIWORD(wParam) == LBN_SELCHANGE)
1068:                   {
1069:                           CheckImportWizButtonStatus(pSetup, hwndDlg);
1070:                   }
1071:                   break;
1072:           }
1073:           break;
1074:
1075:   case WM_NOTIFY:
1076:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1077:           assert(pSetup);
1078:           switch (((NMHDR *)lParam)->code)
1079:           {
1080:           case PSN_SETACTIVE:
1081:                   assert(pSetup->pWeb);
1082:                   pSetup->fNext = FALSE;
1083:
1084:                   switch (pSetup->pWeb->checkOC())
1085:                   {
1086:                   case YAHOO:
1087:                           SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
GetRscString(IDS_WEB_YAHOO_PORTFOLIOS));
1088:                           break;
1089:
1090:                   case QUICKENDC:
1091:                           SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
GetRscString(IDS_WEB_QUICKEN_PORTFOLIOS));
1092:                           break;
1093:
1094:                   default:
1095:                           Assert(FALSE);
1096:                   }
1097:
```

```
1098:                       if(pSetup->pWeb->EnsurePortfolioList(NULL, hwndDlg))
1099:                       {
1100:                               SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST,
LB_RESETCONTENT, 0, 0);
1101:                               PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
1102:                       }
1103:                       else
1104:                       {
1105:                               PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
1106:                               CheckImportWizButtonStatus(pSetup, hwndDlg);
1107:                       }
1108:
1109:                       break;
1110:
1111:               case PSN_WIZBACK:
1112:                       SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB);
1113:                       return TRUE;
1114:
1115:               case PSN_WIZNEXT:
1116:                       // BUG BUG BUG! Should be ensuring an item is selected
before getting here.
1117:                       if (pSetup->pWeb->GetAccountsSelected(hwndDlg))
1118:                       {
1119:                               SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_WEB_FINISH);
1120:                       }
1121:                       else
1122:                       {
1123:                               SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1124:                       }
1125:                       return(TRUE);
1126:
1127:               }
1128:               break;
1129:
1130:       case WM_INITDIALOG:
1131:               pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1132:               SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1133:
1134:               assert(pSetup);
1135:
1136:               return(TRUE);
1137:       }
1138:
1139:       return(FALSE);
1140:}
1141:
1142://-------------------------------------------------------------------------
------------
1143:// Dialog proc for the Account Wizard Yahoo Finish Property Sheet.
1144://-------------------------------------------------------------------------
------------
1145:BOOL CALLBACK AccountWizWebFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1146:{
1147:       SETUP_STRUCT    *pSetup = NULL;
1148:
1149:       switch(uMsg)
1150:       {
1151:       case WM_NOTIFY:
1152:               pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
```

```
1153:           assert(pSetup);
1154:           switch (((NMHDR *)lParam)->code)
1155:           {
1156:           case PSN_SETACTIVE:
1157:                   assert(pSetup->pWeb);
1158:
1159:                   if (pSetup->pWeb->checkOC() == QUICKENDC) {
1160:                           SetDlgItemText(hwndDlg, IDC_DOWNLOAD_CAPTION,
"Downloading Quicken.com information...");
1161:                   }
1162:                   break;
1163:           case PSN_WIZBACK:
1164:                   SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_WEB_LIST);
1165:                   return TRUE;
1166:           }
1167:           break;
1168:   case WM_INITDIALOG:
1169:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1170:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1171:
1172:           assert(pSetup);
1173:
1174:           return(TRUE);
1175:   }
1176:   return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
1177:}
1178:
1179://-------------------------------------------------------------------------
------------
1180:// Dialog proc for the Account Wizard Money 1 Property Sheet.
1181://-------------------------------------------------------------------------
------------
1182:BOOL CALLBACK AccountWizMoney1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1183:{
1184:   SETUP_STRUCT    *pSetup = NULL;
1185:
1186:   switch(uMsg)
1187:   {
1188:   case WM_PAINT:
1189:           PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1190:           break;
1191:
1192:   case WM_COMMAND:
1193:           switch(HIWORD(wParam))
1194:           {
1195:           case BN_DBLCLK:
1196:                   PropSheet_PressButton(GetParent(hwndDlg), PSBTN_NEXT);
1197:                   break;
1198:           }
1199:           break;
1200:
1201:   case WM_NOTIFY:
1202:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1203:           assert(pSetup);
1204:           switch (((NMHDR *)lParam)->code)
1205:           {
1206:           case PSN_SETACTIVE:
1207:                   PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
1208:                   break;
1209:
```

```
1210:           case PSN_WIZNEXT:
1211:                   SW_ACCT_TYPE            swatSelect;
1212:
1213:                   swatSelect = SWAT_MONEY;
1214:
1215:                   if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY_IMPORT))
1216:                           swatSelect = SWAT_MONEY_IMPORT;
1217:
1218:                   if (IsDlgButtonChecked(hwndDlg, IDC_ACCOUNT_MONEY_LINK))
1219:                           swatSelect = SWAT_MONEY_LINK;
1220:
1221:                   switch (swatSelect)
1222:                   {
1223:                   case SWAT_MONEY_IMPORT:
1224:                           pSetup->pMoney->SetLink(FALSE);
1225:                           break;
1226:
1227:                   case SWAT_MONEY_LINK:
1228:                           if (GET_ROAMING(pSetup->pOCX)->GetRoamingState() !=
RMS_OFF)
1229:                           {
1230:
MessageBox(OCX(pSetup->pOCX)->GetInnerWindow(), GetRscString2(IDS_ROAMING_ERR_LINK),
GetRscString(IDS_ERROR), MB_OK | MB_ICONSTOP | MB_APPLMODAL);
1231:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1232:                           }
1233:                           else
1234:                           {
1235:                                   pSetup->pMoney->SetLink(TRUE);
1236:                           }
1237:                           break;
1238:
1239:                   default:
1240:                           assert(FALSE);
1241:                           SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1242:                           break;
1243:                   }
1244:
1245:                   pSetup->swatSelect = swatSelect;
1246:                   return TRUE;
1247:
1248:           case PSN_WIZBACK:
1249:                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
1250:                   return TRUE;
1251:
1252:           }
1253:           break;
1254:
1255:   case WM_INITDIALOG:
1256:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1257:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1258:
1259:           assert(pSetup->pMoney);
1260:
1261:           CheckDlgButton(hwndDlg, IDC_ACCOUNT_MONEY_IMPORT, BST_CHECKED);
1262:
1263:           if (!pSetup->pOCX->IsMoneyInstalled(MT_ANY))
1264:           {
1265:                   EnableControl(hwndDlg, IDC_ACCOUNT_MONEY_LINK, FALSE);
1266:           }
1267:
1268:           return(TRUE);
```

```
1269:    }
1270:
1271:    return(FALSE);
1272:}
1273:
1274://--------------------------------------------------------------------------
------------
1275:// Dialog proc for the Account Wizard Money 2 Property Sheet.
1276://--------------------------------------------------------------------------
------------
1277:BOOL CALLBACK AccountWizMoneyImportDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1278:{
1279:    SETUP_STRUCT    *pSetup = NULL;
1280:
1281:    switch(uMsg)
1282:    {
1283:    case WM_PAINT:
1284:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1285:            break;
1286:
1287:    case WM_COMMAND:
1288:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1289:            switch(LOWORD(wParam))
1290:            {
1291:            case IDC_ACCOUNT_LIST:
1292:                    if (HIWORD(wParam) == LBN_SELCHANGE)
1293:                    {
1294:                            CheckImportWizButtonStatus(pSetup, hwndDlg);
1295:                    }
1296:                    break;
1297:
1298:            case IDC_FILE:
1299:                    if (HIWORD(wParam) == EN_KILLFOCUS)
1300:                    {
1301:                            if (pSetup->fClosing == FALSE)
1302:                            {
1303:                                    // Do the default behavior
1304:                                    MoneyAccountsDialogProc(hwndDlg, uMsg,
wParam, lParam);
1305:
1306:                                    // Check the selection
1307:                                    // CheckImportWizButtonStatus(hwndDlg);
1308:                            }
1309:
1310:                            // Don't pass this message along, because we already
did if necessary.
1311:                            return(FALSE);
1312:                    }
1313:                    break;
1314:
1315:            case IDC_BROWSE:
1316:                    // Do the default behavior
1317:                    MoneyAccountsDialogProc(hwndDlg, uMsg, wParam, lParam);
1318:
1319:                    // Check the selection
1320:                    CheckImportWizButtonStatus(pSetup, hwndDlg);
1321:
1322:                    // Don't pass this message along, because we already did if
necessary.
1323:                    return(FALSE);
1324:            }
```

```
1325:           break;
1326:
1327:   case WM_NOTIFY:
1328:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1329:           switch (((NMHDR *)lParam)->code)
1330:           {
1331:           case PSN_SETACTIVE:
1332:                   pSetup->fClosing = FALSE;
1333:                   pSetup->fNext = FALSE;
1334:
1335:                   PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
1336:
1337:                   pSetup->pMoney->SetReportError(FALSE);
1338:                   pSetup->pMoney->FillAccountList(hwndDlg, NULL);

1339:
1340:                   CheckImportWizButtonStatus(pSetup, hwndDlg);
1341:                   break;
1342:
1343:           case PSN_WIZNEXT:
1344:                   pSetup->pMoney->SetReportError(TRUE);
1345:
1346:                   if (pSetup->pMoney->GetAccountsSelected(hwndDlg))
1347:                   {
1348:                           pSetup->fClosing = TRUE;
1349:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_FINISH);
1350:                   }
1351:                   else
1352:                   {
1353:                           SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1354:                   }
1355:
1356:                   return TRUE;
1357:
1358:           case PSN_WIZBACK:
1359:                   SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_MONEY_1);
1360:                   return TRUE;
1361:           }
1362:           break;
1363:
1364:   case WM_INITDIALOG:
1365:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1366:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1367:
1368:           // ImportAccountsDialogProc handles most of the initialization
1369:           lParam = (LPARAM)pSetup->pMoney;
1370:           break;
1371:   }
1372:
1373:   return(MoneyAccountsDialogProc(hwndDlg, uMsg, wParam, lParam));
1374:}
1375:
1376://----------------------------------------------------------------------------
------------
1377:// Dialog proc for the Account Wizard Money Finish Property Sheet.
1378://----------------------------------------------------------------------------
------------
1379:BOOL CALLBACK AccountWizMoneyFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1380:{
1381:   SETUP_STRUCT    *pSetup = NULL;
1382:
```

```
1383:    switch(uMsg)
1384:    {
1385:    case WM_NOTIFY:
1386:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1387:            assert(pSetup);
1388:            switch (((NMHDR *)lParam)->code)
1389:            {
1390:            case PSN_WIZBACK:
1391:                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MONEY_IMPORT);
1392:                    return TRUE;
1393:            }
1394:            break;
1395:    }
1396:
1397:    return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
1398:}
1399:
1400://----------------------------------------------------------------------------
-----------
1401:// Dialog proc to enable/disable the next button in response to various events
1402://----------------------------------------------------------------------------
-----------
1403:void CheckQuickenButtonStatus(SETUP_STRUCT *pSetup, HWND hwndDlg)
1404:{
1405:    CQuicken *pQuicken = pSetup->pQuicken;
1406:
1407:    if (pQuicken->IsQReadInstalled())
1408:            CheckImportWizButtonStatus(pSetup, hwndDlg);
1409:    else
1410:            PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
1411:}
1412:
1413://----------------------------------------------------------------------------
-----------
1414:// Dialog proc for the Account Wizard Quicken Property Sheet.
1415://----------------------------------------------------------------------------
-----------
1416:BOOL CALLBACK AccountWizQuickenDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1417:{
1418:    SETUP_STRUCT    *pSetup = NULL;
1419:
1420:    switch(uMsg)
1421:    {
1422:    case WM_PAINT:
1423:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1424:            break;
1425:
1426:    case WM_USER_QREAD_DOWNLOAD:
1427:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1428:            CheckQuickenButtonStatus(pSetup, hwndDlg);
1429:            break;
1430:
1431:    case WM_COMMAND:
1432:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1433:            switch(LOWORD(wParam))
1434:            {
1435:            case IDC_ACCOUNT_LIST:
1436:                    if (HIWORD(wParam) == LBN_SELCHANGE)
1437:                    {
1438:                            assert(GetDlgItem(hwndDlg, IDC_ACCOUNT_LIST) ==
```

```
(HWND)lParam);
1439:                                   CheckQuickenButtonStatus(pSetup, hwndDlg);
1440:                           }
1441:                           break;
1442:
1443:                   case IDC_BROWSE:
1444:                           // Do the default behavior
1445:                           QuickenAccountsDialogProc(hwndDlg, uMsg, wParam, lParam);
1446:
1447:                           // Check the selection
1448:                           CheckQuickenButtonStatus(pSetup, hwndDlg);
1449:
1450:                           // Don't pass this message along, because we already did if
necessary.
1451:                           return(FALSE);
1452:                   }
1453:                   break;
1454:
1455:           case WM_NOTIFY:
1456:                   pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1457:                   switch (((NMHDR *)lParam)->code)
1458:                   {
1459:                   case PSN_SETACTIVE:
1460:                           assert(pSetup->pQuicken);
1461:                           pSetup->fNext = FALSE;
1462:                           PropSheet_SetWizButtons (GetParent(hwndDlg), PSWIZB_BACK);
1463:                           pSetup->pQuicken->EnsureQReadDLL(NULL, hwndDlg);
1464:                           CheckQuickenButtonStatus(pSetup, hwndDlg);
1465:                           break;
1466:
1467:                   case PSN_WIZBACK:
1468:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
1469:                           return TRUE;
1470:
1471:                   case PSN_WIZNEXT:
1472:                           pSetup->pQuicken->SetReportError(TRUE);
1473:
1474:                           if (pSetup->pQuicken->GetAccountsSelected(hwndDlg))
1475:                           {
1476:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_QUICKEN_FINISH);
1477:                           }
1478:                           else
1479:                           {
1480:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1481:                           }
1482:                           return TRUE;
1483:                   }
1484:                   break;
1485:
1486:           case WM_INITDIALOG:
1487:                   pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1488:                   SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1489:
1490:                   // ImportAccountsDialogProc handles most of the initialization
1491:                   lParam = (LPARAM)pSetup->pQuicken;
1492:                   break;
1493:           }
1494:
1495:           return(QuickenAccountsDialogProc(hwndDlg, uMsg, wParam, lParam));
1496:}
1497:
```

```
1498://-------------------------------------------------------------------------
-----------
1499:// Dialog proc for the Account Wizard Quciken Finish Property Sheet.
1500://-------------------------------------------------------------------------
-----------
1501:BOOL CALLBACK AccountWizQuickenFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1502:{
1503:    SETUP_STRUCT    *pSetup = NULL;
1504:
1505:    switch(uMsg)
1506:    {
1507:    case WM_NOTIFY:
1508:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1509:            assert(pSetup);
1510:            switch (((NMHDR *)lParam)->code)
1511:            {
1512:            case PSN_WIZBACK:
1513:                    SetWindowLong(hwndDlg, DWL_MSGRESULT, IDD_ACCOUNT_QUICKEN);
1514:                    return TRUE;
1515:
1516:            }
1517:            break;
1518:    }
1519:
1520:    return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
1521:}
1522:
1523://-------------------------------------------------------------------------
-----------
1524:// Dialog proc for the Account Wizard Master Password Property Sheet.
1525://-------------------------------------------------------------------------
-----------
1526:BOOL CALLBACK AccountWizMasterPasswordDialogProc(HWND hwndDlg, UINT uMsg,
WPARAM wParam, LPARAM lParam)
1527:{
1528:    SETUP_STRUCT    *pSetup = NULL;
1529:    TCHAR szTitle[MAX_STRING];
1530:
1531:    switch(uMsg)
1532:    {
1533:    case WM_PAINT:
1534:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1535:            break;
1536:
1537:    case WM_NOTIFY:
1538:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1539:            assert(pSetup);
1540:            switch (((NMHDR *)lParam)->code)
1541:            {
1542:            case PSN_SETACTIVE:
1543:                    if (pSetup->wNAWEntry == NAW_UPDATE_ACCOUNT)
1544:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_NEXT);
1545:                    else
1546:                            PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
1547:                    break;
1548:
1549:            case PSN_WIZBACK:
1550:                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
```

```
1551:                   return TRUE;
1552:
1553:           case PSN_WIZNEXT:
1554:                   if (ValidatePassword(hwndDlg, SEC_FLAG_VERIFY_LENGTH,
IDC_PASSWORD_NEW, IDC_PASSWORD_NEW_CONFIRM))
1555:                   {
1556:
OCX(pSetup->pOCX)->SetLastMPChangeJulian(g_pDateToday->GetJulian());
1557:                           GetDlgItemText(hwndDlg, IDC_PASSWORD_NEW,
pSetup->pOFX->GetPassword(), MAX_STRING);
1558:
OCX(pSetup->pOCX)->SetMasterPassword(pSetup->pOFX->GetPassword());
1559:                           SetFirstOFXPropSheet(hwndDlg, pSetup);
1560:                   }
1561:                   else
1562:                   {
1563:                           SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1564:                   }
1565:                   return TRUE;
1566:           }
1567:           break;
1568:
1569:   case WM_INITDIALOG:
1570:           {
1571:                   pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE
*)lParam)->lParam);
1572:                   SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1573:                   SetDlgItemText(hwndDlg, IDC_PASSWORD_NEW,
pSetup->pOFX->GetPassword());
1574:                   SetDlgItemText(hwndDlg, IDC_PASSWORD_NEW_CONFIRM,
pSetup->pOFX->GetPassword());
1575:
1576:                   // Brand the title
1577:                   if (GetWindowText(hwndDlg, szTitle, MAX_STRING))
1578:                   {
1579:                           OCX(pSetup->pOCX)->ApplyIPKBranding(szTitle, NULL,
MAX_STRING);
1580:                           PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
1581:                   }
1582:
1583:           }
1584:           return(TRUE);
1585:   }
1586:
1587:   return(FALSE);
1588:}
1589:
1590://--------------------------------------------------------------------------
------------
1591:// Activate controls for OFX import
1592://--------------------------------------------------------------------------
------------
1593:void AccountWizOFX1PSNActiveDlgProc(HWND hwndDlg)
1594:{
1595:   DWORD   dwFlags = 0;
1596:
1597:   SETUP_STRUCT *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
1598:   assert(pSetup);
1599:
1600:   // Do we have the possibility of a next button? Check for a listview
selection.
1601:   if (ListView_GetSelectedCount(GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST)) ==
```

```
1)
1602:   {
1603:           dwFlags |= PSWIZB_NEXT;
1604:   }
1605:
1606:   if ((pSetup->wNAWEntry != NAW_UPDATE_ACCOUNT)
1607:           || ((pSetup->pOCX->GetSecurityLevel() != SECURITY_HIGH)
1608:           && (FileHasEncryptedOFXData(pSetup->pOCX) == FALSE)))
1609:   {
1610:           dwFlags |= PSWIZB_BACK;
1611:   }
1612:
1613:   PropSheet_SetWizButtons(GetParent(hwndDlg), dwFlags);
1614:}
1615:
1616://-------------------------------------------------------------------------
------------
1617:// Dialog proc for the Account Wizard OFX Property Sheet #1.
1618://-------------------------------------------------------------------------
----------
1619:BOOL CALLBACK AccountWizOFX1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam,
LPARAM lParam)
1620:{
1621:   SETUP_STRUCT            *pSetup = NULL;
1622:   HWND                            hwndList;
1623:
1624:   switch(uMsg)
1625:   {
1626:   case WM_USER_PARSER_DOWNLOAD:
1627:           AccountWizOFX1PSNActiveDlgProc(hwndDlg);
1628:           break;
1629:
1630:   case WM_USER_BROKERLIST_DOWNLOAD:
1631:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1632:           assert(pSetup);
1633:
1634:           hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);
1635:
1636:           // Get bitmaps
1637:           pSetup->pOFX->EnsureBrokerImages(hwndList);
1638:
1639:           pSetup->pOFX->FillBrokerList(hwndList);
1640:
1641:           EnableWindow(hwndList, TRUE);
1642:
1643:           AccountWizOFX1PSNActiveDlgProc(hwndDlg);
1644:           break;
1645:
1646:   case WM_USER_BROKERIMAGE_DOWNLOAD:
1647:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1648:           assert(pSetup);
1649:
1650:           hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);
1651:
1652:           pSetup->pOFX->UpdateBrokerList(hwndList);
1653:
1654:           pSetup->pOFX->EnsureBrokerImages(hwndList);
1655:           break;
1656:
1657:   case WM_PAINT:
1658:           PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
1659:           break;
```

```
1660:
1661:    case WM_NOTIFY:
1662:            if (wParam == IDC_OFX_BROKER_LIST)
1663:            {
1664:                    NM_LISTVIEW     *pLV = (NM_LISTVIEW *)lParam;
1665:
1666:                    switch (pLV->hdr.code)
1667:                    {
1668:                    case NM_DBLCLK: // same as an OK
1669:                            PropSheet_PressButton(GetParent(hwndDlg),
PSBTN_NEXT);
1670:                            break;
1671:
1672:                    case LVN_ITEMCHANGED:
1673:                            if (pLV->uNewState & LVIS_SELECTED)
1674:                                    AccountWizOFX1PSNActiveDlgProc(hwndDlg);
1675:                            break;
1676:                    }
1677:            }
1678:            else
1679:            {
1680:                    pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
1681:                    assert(pSetup);
1682:
1683:                    switch (((NMHDR *)lParam)->code)
1684:                    {
1685:                    case PSN_QUERYCANCEL:
1686:
GET_ERROR(pSetup->pOCX)->DisplayNotification(GetParent(hwndDlg),
1687:
                            IDS_MORE_BROKERS_COMING,
1688:
                            IDS_PORTFOLIO_MANAGER,
1689:
                            MB_ICONEXCLAMATION, MB_OK);
1690:                            return (FALSE);         // FALSE means accept the
cancel
1691:
1692:                    case PSN_SETACTIVE:
1693:                            assert(pSetup->pOFX);
1694:
1695:                            // Read broker.txt.
1696:                            pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg);
1697:
1698:                            pSetup->pOFX->EnsureOFXParser(NULL);
1699:
1700:                            AccountWizOFX1PSNActiveDlgProc(hwndDlg);
1701:                            break;
1702:
1703:                    case PSN_WIZBACK:
1704:                            if (FileHasEncryptedOFXData(pSetup->pOCX)
1705:                                    || (pSetup->pOCX->GetSecurityLevel() ==
SECURITY_HIGH))
1706:                            {
1707:                                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
1708:                            }
1709:                            else
1710:                                    SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_MASTER_PASSWORD);
1711:
1712:                            return TRUE;
```

```
1713:
1714:                   case PSN_WIZNEXT:
1715:                           COFXSession     *pOFXSession;
1716:
1717:                           hwndList = GetDlgItem(hwndDlg, IDC_OFX_BROKER_LIST);
1718:
1719:                           if (ListView_GetSelectedCount(hwndList) == 0)
1720:                           {
1721:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
1722:                                   return(TRUE);
1723:                           }
1724:
1725:                           pOFXSession = pSetup->pOFX->GetCurSession();
1726:                           if (pOFXSession !=
pSetup->pOFX->SetCurSessionToSelection(hwndList))
1727:                           {
1728:                                   pOFXSession = pSetup->pOFX->GetCurSession();
1729:
1730:                                   // Broker selection changed, cancel the
previous profile download
1731:                                   //  if one is in progress.
1732:                                   if (pSetup->pOFXDownload)
1733:                                   {
1734:                                           // CancelDownload releases the
object, and our callback zeros out pOFXDownload
1735:
pSetup->pOFXDownload->CancelDownload();
1736:
ReleaseInterface(pSetup->pOFXDownload);
1737:                                   }
1738:                           }
1739:
1740:                           if (pOFXSession->fNotGrouping())
1741:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_NOGROUPING);
1742:                           else
1743:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
1744:                           return TRUE;
1745:                   }
1746:           }
1747:           break;
1748:
1749:   case WM_INITDIALOG:
1750:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
1751:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
1752:
1753:           if ((pSetup) && (pSetup->pOFX))
1754:                   pSetup->pOFX->SetHWndParent(hwndDlg);
1755:
1756:           return(TRUE);
1757:   }
1758:
1759:   return(FALSE);
1760:}
1761:
1762://-------------------------------------------------------------------------
------------
1763:// Change the dialog text and controls now that we have profile information
1764://-------------------------------------------------------------------------
------------
1765:void AccountWizOFXNavigationDlgProc(HWND hwndDlg, SETUP_STRUCT *pSetup, long
idcText1, long idcText2)
```

```
1766:{
1767:   COFXSession     *pOFXSession = pSetup->pOFX->GetCurSession();
1768:   DWORD           dwWizButtons = 0;
1769:
1770:   // if this is not being updated, or we are updating but we didn't know the
broker
1771:   //  before we updated, we have a back button.
1772:   if ((pSetup->wNAWEntry != NAW_UPDATE_ACCOUNT) || ((pSetup->wNAWModifier &
NAW_MODIFIER_NEW_ACCOUNT) != 0))
1773:           dwWizButtons |= PSWIZB_BACK;
1774:
1775:   // For the next button to be activated, both text fields must have text, and
the profile must be downloaded.
1776:   if (pOFXSession && pOFXSession->GetProfile() && DoesEditBoxHaveText(hwndDlg,
idcText1) && DoesEditBoxHaveText(hwndDlg, idcText2))
1777:           dwWizButtons |= PSWIZB_NEXT;
1778:
1779:   PropSheet_SetWizButtons (GetParent(hwndDlg), dwWizButtons);
1780:}
1781:
1782://-------------------------------------------------------------------------
-----------
1783:// Change the dialog text and controls with custom text from broker.txt
1784://-------------------------------------------------------------------------
-----------
1785:void AccountWizOFXCustomTextDlgProc(HWND hwndDlg)
1786:{
1787:   SETUP_STRUCT    *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
1788:   COFXSession             *pOFXSession;
1789:   OFX_BROKER              *pBroker;
1790:
1791:   assert(pSetup);
1792:   assert(pSetup->pOFX);
1793:
1794:   pOFXSession = pSetup->pOFX->GetCurSession();
1795:   pBroker = pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());
1796:
1797:   // Prompts
1798:   if (pBroker && pBroker->szUserIDPrompt[0])
1799:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_USERID_CAPTION,
pBroker->szUserIDPrompt);
1800:   else
1801:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_USERID_CAPTION,
GetRscString(IDS_OFX_USERID_DEFAULT));
1802:
1803:   if (pBroker && pBroker->szPasswordPrompt[0])
1804:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_PASSWORD_CAPTION,
pBroker->szPasswordPrompt);
1805:   else
1806:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_PASSWORD_CAPTION,
GetRscString(IDS_OFX_PASSWORD_DEFAULT));
1807:
1808:   if (pBroker && pBroker->szAccountPrompt[0])
1809:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER_CAPTION,
pBroker->szAccountPrompt);
1810:   else
1811:           SetDlgItemText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER_CAPTION,
GetRscString(IDS_OFX_ACCOUNT_NUMBER_DEFAULT));
1812:
1813:   // Informational text items
1814:   if (pBroker && pBroker->szSignupInfo[0])
1815:           SetDlgItemText(hwndDlg, IDC_OFX_SIGNUP_INFO, pBroker->szSignupInfo);
```

```
1816:   else
1817:           SetDlgItemText(hwndDlg, IDC_OFX_SIGNUP_INFO,
GetRscString(IDS_OFX_SIGNUP_INFO_DEFAULT));
1818:
1819:   if (pBroker && pBroker->szCustomInfo)
1820:           SetDlgItemText(hwndDlg, IDC_OFX_CUSTOM_INFO, pBroker->szCustomInfo);
1821:
1822:   if (pBroker && pBroker->szAccountInfo[0])
1823:           SetDlgItemText(hwndDlg, IDC_OFX_ACCOUNT_INFO,
pBroker->szAccountInfo);
1824:   else
1825:           SetDlgItemText(hwndDlg, IDC_OFX_ACCOUNT_INFO,
GetRscString(IDS_OFX_ACCOUNT_INFO_DEFAULT));
1826:}
1827:
1828:
1829:
1830://------------------------------------------------------------------------
------------
1831:// Change the dialog text and controls now that we have profile information
1832://------------------------------------------------------------------------
------------
1833:void AccountWizOFXProfileActiveDlgProc(HWND hwndDlg)
1834:{
1835:   SETUP_STRUCT    *pSetup = (SETUP_STRUCT *)GetProp(hwndDlg,
SETUPPROP_PROP_NAME);
1836:   COFXSession             *pOFXSession = pSetup->pOFX->GetCurSession();
1837:
1838:   assert(pSetup);
1839:   assert(pOFXSession);
1840:}
1841:
1842:
1843://------------------------------------------------------------------------
------------
1844:// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
1845://------------------------------------------------------------------------
------------
1846:BOOL GetBrokerProfileInformation(COFXSession *pOFXSession, HWND hwndDlg,
SETUP_STRUCT *pSetup)
1847:{
1848:   BOOL    fResult = FALSE;
1849:
1850:   assert(pOFXSession);
1851:   assert(hwndDlg);
1852:   assert(pSetup);
1853:
1854:   // Don't toss alerts if this dialog is running a download.
1855:   //  Check by seeing if the progress bar is currently visible.
1856:   if (ShowWindow(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS), SW_SHOWNA) == 0)
1857:   // if (!IsWindowVisible(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS)))
1858:   {
1859:           COFXProfile     *pProfile = new COFXProfile(pSetup->pOCX, hwndDlg,
pSetup->pOFX->GetOFXCom());
1860:
1861:           pOFXSession->SetLParam((LPARAM)hwndDlg);
1862:           fResult = pProfile->InitiateProfile(pOFXSession);
1863:
1864:           // We should have cancelled any pending profile download already.
1865:           assert(pSetup->pOFXDownload == NULL);
1866:           pSetup->pOFXDownload = pOFXSession->GetDownload();
1867:           pSetup->pOFXDownload->AddRef();
1868:   }
```

```
1869:
1870:   return(fResult);
1871:}
1872:
1873://----------------------------------------------------------------------------
------------
1874:// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
1875://----------------------------------------------------------------------------
------------
1876:BOOL AccountWizBrokerListDlgProc(HWND hwndDlg, SETUP_STRUCT *pSetup, long
idsTitle, long idcText1, long idcText2)
1877:{
1878:   BOOL                    fSuccess = FALSE;
1879:   COFXSession             *pOFXSession;
1880:
1881:   assert(pSetup);
1882:
1883:   pOFXSession = pSetup->pOFX->GetCurSession();
1884:
1885:   if (pOFXSession)
1886:   {
1887:           char    szTitle[MAX_STRING];
1888:
1889:           AccountWizOFXNavigationDlgProc(hwndDlg, pSetup, idcText1, idcText2);
1890:
1891:           if (!pOFXSession->GetProfile())
1892:           {
1893:                   fSuccess = GetBrokerProfileInformation(pOFXSession, hwndDlg,
pSetup);
1894:           }
1895:           else
1896:           {
1897:                   fSuccess = TRUE;
1898:                   AccountWizOFXProfileActiveDlgProc(hwndDlg);
1899:           }
1900:
1901:           // Set the prop sheet title
1902:           wsprintf(szTitle, GetRscString(idsTitle),
pOFXSession->GetServerName());
1903:           PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
1904:   }
1905:
1906:   return(fSuccess);
1907:}
1908:
1909:
1910://----------------------------------------------------------------------------
------------
1911:// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
1912://----------------------------------------------------------------------------
------------
1913:COFXSession *SetCurSessionToIPKBroker(SETUP_STRUCT *pSetup)
1914:{
1915:   COFXSession     *pOFXSession = NULL;
1916:
1917:   if (pSetup->pOCX->GetIPKBrokerID() != NULL_BROKER)
1918:   {
1919:           OFX_BROKER *pBroker =
pSetup->pOFX->FindBroker(pSetup->pOCX->GetIPKBrokerID());
1920:           assert(pBroker);
1921:
1922:           pOFXSession = pSetup->pOFX->SetCurSessionToBroker(pBroker);
1923:   }
```

```
1924:
1925:    return(pOFXSession);
1926:}
1927:
1928:
1929://----------------------------------------------------------------------------
-----------
1930:// Dialog proc for the Account Wizard OFX No Account Grouping Property Sheet .
1931://----------------------------------------------------------------------------
-----------
1932:BOOL CALLBACK AccountWizOFXNoGroupingDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
1933:{
1934:    SETUP_STRUCT    *pSetup = NULL;
1935:
1936:    switch(uMsg)
1937:    {
1938:    case WM_USER_SETFOCUS:
1939:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1940:            DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
1941:            break;
1942:
1943:    case WM_USER_PARSER_DOWNLOAD:
1944:            break;
1945:
1946:    case WM_USER_BROKERLIST_DOWNLOAD:
1947:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1948:            assert(pSetup);
1949:
1950:            if (pSetup->pOCX->IsIPK())
1951:                    SetCurSessionToIPKBroker(pSetup);
1952:
1953:            {
1954:                    COFXSession *pOFXSession = pSetup->pOFX->GetCurSession();
1955:                    OFX_BROKER *pBroker =
pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());
1956:
1957:                    // Get bitmaps
1958:                    pSetup->pOFX->DownloadBrokerImage(pBroker);
1959:            }
1960:
1961:            AccountWizBrokerListDlgProc(hwndDlg, pSetup, IDS_OFX_ACCOUNT_TITLE,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
1962:            break;
1963:
1964:    case WM_USER_OFX_DOWNLOAD:
1965:            pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1966:            ReleaseInterface(pSetup->pOFXDownload);
1967:
1968:            AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
1969:            AccountWizOFXProfileActiveDlgProc(hwndDlg);
1970:            break;
1971:
1972:    case WM_USER_BROKERIMAGE_DOWNLOAD:
1973:            // intentionally repaint bitmap.
1974:    case WM_PAINT:
1975:            if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
1976:                    PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
1977:
1978:            PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
```

```
1979:           break;
1980:
1981:   case WM_COMMAND:
1982:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1983:           switch(HIWORD(wParam))
1984:           {
1985:           case EN_CHANGE:
1986:                   AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
1987:                   break;
1988:           }
1989:           break;
1990:
1991:   case WM_NOTIFY:
1992:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
1993:           assert(pSetup);
1994:           switch (((NMHDR *)lParam)->code)
1995:           {
1996:           case PSN_SETACTIVE:
1997:                   assert(pSetup->pOFX);
1998:
1999:                   AccountWizOFXCustomTextDlgProc(hwndDlg);
2000:
2001:                   pSetup->pOFX->EnsureOFXParser(NULL);
2002:
2003:                   if (pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg))
2004:                   {
2005:                           // We have the list already, do normal activation.
2006:                           AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_ACCOUNT_TITLE, IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
2007:                   }
2008:                   break;
2009:
2010:           case PSN_WIZBACK:
2011:                   // Grab dialog info before we leave in case we come back
2012:                   pSetup->pOFX->GetAccountData(hwndDlg, 0,
IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);
2013:
2014:                   // IPK skips over the dialog to select a broker
2015:                   if (pSetup->pOCX->GetIPKBrokerID() == NULL_BROKER)
2016:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_1);
2017:                   else
2018:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
2019:                   return(TRUE);
2020:
2021:           case PSN_WIZNEXT:
2022:                   pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);
2023:
2024:                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
2025:                   return TRUE;
2026:           }
2027:           break;
2028:
2029:   case WM_INITDIALOG:
2030:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
2031:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
2032:
2033:           assert(pSetup->pOFX);
2034:           if ((pSetup) && (pSetup->pOFX))
```

```
2035:           {
2036:                   pSetup->pOFX->SetAccountData(hwndDlg, 0,
IDC_ACCOUNT_OFX_PASSWORD, IDC_ACCOUNT_OFX_NUMBER);
2037:                   pSetup->pOFX->SetHWndParent(hwndDlg);
2038:           }
2039:
2040:           return(TRUE);
2041:   }
2042:
2043:   return(FALSE);
2044:}
2045:
2046:
2047://-------------------------------------------------------------------------
------------
2048:// Dialog proc for the Account Wizard OFX Grouping 1 Property Sheet.
2049://-------------------------------------------------------------------------
------------
2050:BOOL CALLBACK AccountWizOFXGrouping1DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
2051:{
2052:   SETUP_STRUCT    *pSetup = NULL;
2053:   COFXSession             *pOFXSession;
2054:
2055:   switch(uMsg)
2056:   {
2057:   case WM_USER_SETFOCUS:
2058:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2059:           DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
2060:           break;
2061:
2062:   case WM_USER_PARSER_DOWNLOAD:
2063:           break;
2064:
2065:   case WM_USER_BROKERLIST_DOWNLOAD:
2066:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2067:           assert(pSetup);
2068:
2069:           if (pSetup->pOCX->IsIPK())
2070:                   SetCurSessionToIPKBroker(pSetup);
2071:
2072:           {
2073:                   COFXSession *pOFXSession = pSetup->pOFX->GetCurSession();
2074:                   OFX_BROKER *pBroker;
2075:
2076:                   assert(pOFXSession);
2077:
2078:                   pSetup->pOFX->EnsureSessionParams(pOFXSession);
2079:
2080:                   pBroker =
pSetup->pOFX->FindBroker(pOFXSession->GetBrokerID());
2081:                   assert(pBroker);
2082:
2083:                   if (pOFXSession->fNotGrouping())
2084:                   {
2085:                           // Need to switch to NoGroupingDialog. This can
happen through the
2086:                           //  Account details dialog if the user clicks the
setup button
2087:                           //  to access OFX account information after setting
up a
2088:                           //  Non-grouping OFX account.
2089:                           PropSheet_SetCurSelByID(GetParent(hwndDlg),
```

```
IDD_ACCOUNT_OFX_NOGROUPING);
2090:                   }
2091:                   else
2092:                   {
2093:                           // Get bitmaps
2094:                           pSetup->pOFX->DownloadBrokerImage(pBroker);
2095:
2096:                           AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_ACCOUNT_TITLE, IDC_ACCOUNT_OFX_NUMBER, IDC_ACCOUNT_OFX_PASSWORD);
2097:                   }
2098:           }
2099:
2100:           break;
2101:
2102:   case WM_USER_OFX_DOWNLOAD:
2103:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2104:           ReleaseInterface(pSetup->pOFXDownload);
2105:
2106:           AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
2107:           AccountWizOFXProfileActiveDlgProc(hwndDlg);
2108:           break;
2109:
2110:   case WM_USER_BROKERIMAGE_DOWNLOAD:
2111:           // intentionally repaint bitmap.
2112:   case WM_PAINT:
2113:           if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
2114:                   PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
2115:
2116:           PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
2117:           break;
2118:
2119:   case WM_COMMAND:
2120:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2121:           switch(HIWORD(wParam))
2122:           {
2123:           case EN_CHANGE:
2124:                   AccountWizOFXNavigationDlgProc(hwndDlg, pSetup,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
2125:                   break;
2126:           }
2127:           break;
2128:
2129:   case WM_NOTIFY:
2130:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2131:           assert(pSetup);
2132:           switch (((NMHDR *)lParam)->code)
2133:           {
2134:           case PSN_SETACTIVE:
2135:                   assert(pSetup->pOFX);
2136:
2137:                   AccountWizOFXCustomTextDlgProc(hwndDlg);
2138:
2139:                   pSetup->pOFX->EnsureOFXParser(NULL);
2140:
2141:                   if (pSetup->pOFX->EnsureBrokerList(NULL, hwndDlg))
2142:                   {
2143:                           // We have the list already, do normal activation.
2144:                           AccountWizBrokerListDlgProc(hwndDlg, pSetup,
IDS_OFX_USER_TITLE, IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD);
2145:                   }
```

```
2146:
2147:                   break;
2148:
2149:           case PSN_WIZBACK:
2150:                   // Save data
2151:                   pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);
2152:
2153:                   // IPK skips over the dialog to select a broker
2154:                   if (pSetup->pOCX->GetIPKBrokerID() == NULL_BROKER)
2155:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_1);
2156:                   else
2157:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_IMPORT_EXISTING);
2158:
2159:                   return(TRUE);
2160:
2161:           case PSN_WIZNEXT:
2162:                   pSetup->pOFX->GetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);
2163:
2164:                   pOFXSession = pSetup->pOFX->GetCurSession();
2165:                   assert(pOFXSession);
2166:
2167:                   {
2168:                           COFXSignup      *pSignup = new
COFXSignup(pSetup->pOCX, hwndDlg, pSetup->pOFX->GetOFXCom());
2169:
2170:                           if (pSignup->ValidateProfile(pOFXSession))
2171:                           {
2172:                                   // TODO: Need to check if the FI supports
ACCTINFO.
2173:                                   if (pSignup->GetAvailAccts())
2174:                                           SetWindowLong(hwndDlg,
DWL_MSGRESULT, IDD_ACCOUNT_OFX_GROUPING_3);
2175:                                   else
2176:                                           SetWindowLong(hwndDlg,
DWL_MSGRESULT, IDD_ACCOUNT_OFX_GROUPING_2);
2177:                           }
2178:                           else
2179:                           {
2180:                                   assert(FALSE);  // Broker profile is bad!
2181:                                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_2);
2182:                           }
2183:
2184:                           DeleteInterface(pSignup);
2185:                   }
2186:
2187:                   return(TRUE);
2188:
2189:           }
2190:           break;
2191:
2192:   case WM_INITDIALOG:
2193:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
2194:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
2195:
2196:           assert(pSetup);
2197:           if (pSetup && pSetup->pOFX)
2198:           {
2199:                   pSetup->pOFX->SetHWndParent(hwndDlg);
```

```
2200:
2201:                       pSetup->pOFX->SetAccountData(hwndDlg,
IDC_ACCOUNT_OFX_USERID, IDC_ACCOUNT_OFX_PASSWORD, 0);
2202:               }
2203:
2204:               return(TRUE);
2205:       }
2206:
2207:       return(FALSE);
2208:}
2209:
2210://-------------------------------------------------------------------------
------------
2211:// Dialog proc for the Account Wizard OFX Grouping 2 Property Sheet.
2212://-------------------------------------------------------------------------
------------
2213:BOOL CALLBACK AccountWizOFXGrouping2DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
2214:{
2215:       SETUP_STRUCT    *pSetup = NULL;
2216:       COFXSession             *pOFXSession;
2217:
2218:       switch(uMsg)
2219:       {
2220:       case WM_USER_SETFOCUS:
2221:               pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2222:               DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
2223:               break;
2224:
2225:       case WM_PAINT:
2226:               if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
2227:                       PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
2228:
2229:               PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
2230:               break;
2231:
2232:       case WM_COMMAND:
2233:               switch(HIWORD(wParam))
2234:               {
2235:               case EN_CHANGE:
2236:                       if (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER))
2237:                               PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
2238:                       else
2239:                               PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
2240:                       break;
2241:               }
2242:               break;
2243:
2244:       case WM_NOTIFY:
2245:               pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2246:               assert(pSetup);
2247:               switch (((NMHDR *)lParam)->code)
2248:               {
2249:               case PSN_SETACTIVE:
2250:                       AccountWizOFXCustomTextDlgProc(hwndDlg);
2251:
2252:                       if (DoesEditBoxHaveText(hwndDlg, IDC_ACCOUNT_OFX_NUMBER))
2253:                               PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
```

```
2254:                   else
2255:                           PropSheet_SetWizButtons (GetParent(hwndDlg),
PSWIZB_BACK);
2256:
2257:                   pOFXSession = pSetup->pOFX->GetCurSession();
2258:
2259:                   if (pOFXSession)
2260:                   {
2261:                           TCHAR   szInfo[MAX_STRING];
2262:
2263:                           wsprintf(szInfo,
GetRscString(IDS_OFX_ACCOUNT_TITLE), pOFXSession->GetServerName());
2264:                           PropSheet_SetTitle(GetParent(hwndDlg), 0, szInfo);
2265:                   }
2266:                   break;
2267:
2268:           case PSN_WIZBACK:
2269:                   pSetup->pOFX->GetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);
2270:
2271:                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
2272:                   return(TRUE);
2273:
2274:           case PSN_WIZNEXT:
2275:                   pSetup->pOFX->GetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);
2276:
2277:                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
2278:                   return(TRUE);
2279:
2280:           }
2281:           break;
2282:
2283:   case WM_INITDIALOG:
2284:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
2285:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
2286:
2287:           if (pSetup && pSetup->pOFX)
2288:           {
2289:                   pSetup->pOFX->SetHWndParent(hwndDlg);
2290:
2291:                   // Prefill dialog items if we have account data...
2292:                   pSetup->pOFX->SetAccountData(hwndDlg, 0, 0,
IDC_ACCOUNT_OFX_NUMBER);
2293:           }
2294:
2295:           return(TRUE);
2296:   }
2297:
2298:   return(FALSE);
2299:}
2300:
2301:
2302://-------------------------------------------------------------------------
------------
2303:// Execute an ACCTINFO OFX download to get the list of accounts.
2304://-------------------------------------------------------------------------
------------
2305:BOOL GetBrokerAccountInformation(COFXSession *pOFXSession, HWND hwndDlg,
SETUP_STRUCT *pSetup)
2306:{
```

```
2307:   BOOL            fResult = FALSE;
2308:
2309:   assert(pOFXSession);
2310:   assert(hwndDlg);
2311:   assert(pSetup);
2312:
2313:   // Don't toss alerts if this dialog is running a download.
2314:   //  Check by seeing if the progress bar is currently visible.
2315:   if (ShowWindow(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS), SW_SHOWNA) == 0)
2316:   // if (!IsWindowVisible(GetDlgItem(hwndDlg, IDC_DOWNLOAD_PROGRESS)))
2317:   {
2318:           COFXSignup      *pSignup = new COFXSignup(pSetup->pOCX, hwndDlg,
pSetup->pOFX->GetOFXCom());
2319:
2320:           // Clear the list
2321:           SendDlgItemMessage(hwndDlg, IDC_ACCOUNT_LIST, LB_RESETCONTENT, 0,
0);
2322:
2323:           // Start Account Info download
2324:           pOFXSession->SetLParam((LPARAM)hwndDlg);
2325:           fResult = pSignup->InitiateSignup(pOFXSession);
2326:
2327:           // We should have cancelled any pending profile download already.
2328:           assert(pSetup->pOFXDownload == NULL);
2329:           pSetup->pOFXDownload = pOFXSession->GetDownload();
2330:           pSetup->pOFXDownload->AddRef();
2331:   }
2332:
2333:   return(fResult);
2334:}
2335:
2336://----------------------------------------------------------------------------
------------
2337:// Dialog proc for the Account Wizard OFX Grouping 3 Property Sheet.
2338://----------------------------------------------------------------------------
------------
2339:BOOL CALLBACK AccountWizOFXGrouping3DialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
2340:{
2341:   SETUP_STRUCT    *pSetup = NULL;
2342:   COFXSession             *pOFXSession;
2343:
2344:   switch(uMsg)
2345:   {
2346:   case WM_USER_SETFOCUS:
2347:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2348:           DisplayErrorAndPositionCursor(hwndDlg, wParam, lParam);
2349:           break;
2350:
2351:   case WM_USER_OFX_DOWNLOAD:
2352:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2353:           ReleaseInterface(pSetup->pOFXDownload);
2354:
2355:           // Populate the list with the accounts.
2356:           if (pSetup->pOFX->FillAccountList(hwndDlg, NULL))
2357:           {
2358:                   // Update the navigation elements.
2359:                   PropSheet_SetWizButtons(GetParent(hwndDlg), PSWIZB_BACK |
PSWIZB_NEXT);
2360:           }
2361:
2362:           break;
2363:
```

```
2364:   case WM_PAINT:
2365:           if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
2366:                   PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
2367:
2368:           PaintDlgBmp(hwndDlg, IDB_SETUP, NULL, nDlgBmpLeft, nDlgBmpTop,
nDlgBmpWidth, nDlgBmpHeight, TRUE);
2369:           break;
2370:
2371:   case WM_NOTIFY:
2372:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2373:           assert(pSetup);
2374:           switch (((NMHDR *)lParam)->code)
2375:           {
2376:           case PSN_SETACTIVE:
2377:                   pOFXSession = pSetup->pOFX->GetCurSession();
2378:                   assert(pOFXSession);
2379:
2380:                   if (pOFXSession)
2381:                   {
2382:                           TCHAR   szTitle[MAX_STRING];
2383:
2384:                           // If HasAccounts returns TRUE, we have already
downloaded the accounts
2385:                           //  for this session.
2386:                           if (pOFXSession->GetHasAccounts() == FALSE)
2387:                           {
2388:                                   PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK);
2389:
2390:                                   GetBrokerAccountInformation(pOFXSession,
hwndDlg, pSetup);
2391:                           }
2392:                           else
2393:                           {
2394:                                   PropSheet_SetWizButtons(GetParent(hwndDlg),
PSWIZB_BACK | PSWIZB_NEXT);
2395:                           }
2396:
2397:                           // Set the propsheet title
2398:                           wsprintf(szTitle,
GetRscString(IDS_OFX_SELECT_TITLE), pOFXSession->GetServerName());
2399:                           PropSheet_SetTitle(GetParent(hwndDlg), 0, szTitle);
2400:                   }
2401:                   break;
2402:
2403:           case PSN_WIZBACK:
2404:                   SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_1);
2405:                   return(TRUE);
2406:
2407:           case PSN_WIZNEXT:
2408:                   pSetup->pOFX->SetReportError(TRUE);
2409:
2410:                   if (pSetup->pOFX->GetAccountsSelected(hwndDlg))
2411:                   {
2412:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_FINISH);
2413:                   }
2414:                   else
2415:                   {
2416:                           SetWindowLong(hwndDlg, DWL_MSGRESULT, -1);
2417:                   }
```

```
2418:                   return(TRUE);
2419:
2420:           }
2421:           break;
2422:
2423:   case WM_INITDIALOG:
2424:           pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
2425:           SetProp(hwndDlg, SETUPPROP_PROP_NAME, (void *)pSetup);
2426:
2427:           if ((pSetup) && (pSetup->pOFX))
2428:           {
2429:                   pSetup->pOFX->SetHWndParent(hwndDlg);
2430:
2431:                   if (pSetup->pOFX->GetCurAccount() == NULL)
2432:                   {
2433:                           HWND    hwndList;
2434:
2435:                           hwndList = GetDlgItem(hwndDlg, IDC_ACCOUNT_LIST);
2436:
2437:                           ULONG ulStyle = GetWindowLong(hwndList, GWL_STYLE);
2438:                           ulStyle |= LBS_MULTIPLESEL;
2439:                           SetWindowLong(hwndList, GWL_STYLE, ulStyle);
2440:                   }
2441:           }
2442:
2443:           return(TRUE);
2444:   }
2445:
2446:   return(FALSE);
2447:}
2448:
2449://----------------------------------------------------------------------------
------------
2450:// Dialog proc for the Account Wizard OFX Finish Property Sheet.
2451://----------------------------------------------------------------------------
------------
2452:BOOL CALLBACK AccountWizOFXFinishDialogProc(HWND hwndDlg, UINT uMsg, WPARAM
wParam, LPARAM lParam)
2453:{
2454:   SETUP_STRUCT    *pSetup = NULL;
2455:   COFXSession             *pOFXSession;
2456:
2457:   switch(uMsg)
2458:   {
2459:   case WM_PAINT:
2460:           if (pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME))
2461:                   PaintDlgBmp(hwndDlg, 0, pSetup->pOFX->GetBrandingBitmap(),
nDlgBrandLeft, nDlgBrandTop, nDlgBrandWidth, nDlgBrandHeight, TRUE);
2462:           break;
2463:
2464:   case WM_NOTIFY:
2465:           pSetup = (SETUP_STRUCT *)GetProp(hwndDlg, SETUPPROP_PROP_NAME);
2466:           assert(pSetup);
2467:           switch (((NMHDR *)lParam)->code)
2468:           {
2469:           case PSN_WIZBACK:
2470:                   pOFXSession = pSetup->pOFX->GetCurSession();
2471:
2472:                   if (pOFXSession->fNotGrouping())
2473:                           SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_NOGROUPING);
2474:                   else
2475:                   {
```

```
                                    setupwizlines
2476:                               if (pOFXSession->GetHasAccounts())
2477:                                       SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_3);
2478:                               else
2479:                                       SetWindowLong(hwndDlg, DWL_MSGRESULT,
IDD_ACCOUNT_OFX_GROUPING_2);
2480:                       }
2481:                       return TRUE;
2482:
2483:               case PSN_WIZFINISH:
2484:                       pOFXSession = pSetup->pOFX->GetCurSession();
2485:
2486:                       if (pOFXSession->GetHasAccounts())
2487:                       {
2488:                               pSetup->pOFX->TrimAccountPlex();
2489:                       }
2490:
2491:                       // For Grouping2 case, this flag was never set.
2492:                       pOFXSession->PutHasAccounts(TRUE);
2493:
2494:                       break;
2495:               }
2496:               break;
2497:
2498:       case WM_INITDIALOG:
2499:               pSetup = (SETUP_STRUCT *)(((INVPROPSHEETPAGE *)lParam)->lParam);
2500:               if (pSetup && pSetup->pOCX)
2501:                       pSetup->pOCX->ApplyIPKBranding(hwndDlg,
IDC_OFX_FINISH_TEXT1);
2502:               break;
2503:
2504:       }
2505:
2506:       return(AccountWizFinishDialogProc(hwndDlg, uMsg, wParam, lParam));
2507:}
2508:
2509:
2510://-------------------------------------------------------------------------
-----------
2511:// The beginnings of Investor's tip wizard...
2512://
2513:// Dialog proc for Tip Wizard dialog. Returns TRUE if you should turn off the
tip, FALSE
2514:// if the tip should appear again next time.
2515://-------------------------------------------------------------------------
-----------
2516:BOOL CALLBACK TipDialogProc(HWND hwndDlg, UINT uMsg, WPARAM wParam, LPARAM
lParam)
2517:{
2518:       HBITMAP hBmpTip = NULL;
2519:       RECT rect;
2520:
2521:       switch(uMsg)
2522:       {
2523:       case WM_PAINT:
2524:               {
2525:                       int iTip = (int)GetProp(hwndDlg, TIP_ID_PROP);
2526:                       GetCtlRect(GetDlgItem(hwndDlg, IDC_BUY_BITMAP), hwndDlg,
&rect); // client coordinates
2527:                       PaintDlgBmp(hwndDlg, rgTips[iTip].residBitmap, NULL,
rect.left, rect.top,
2528:                               rect.right - rect.left, rect.bottom - rect.top,
TRUE);
```

```
2529:            }
2530:            break;
2531:
2532:    case WM_COMMAND:
2533:            switch(LOWORD(wParam))
2534:            {
2535:            case IDOK:
2536:                    EndDialog(hwndDlg, IsDlgButtonChecked(hwndDlg,
IDC_TIP_NOT_AGAIN));
2537:                    break;
2538:            }
2539:            break;
2540:
2541:    case WM_INITDIALOG:
2542:            SetDlgItemText(hwndDlg, IDC_TIP_TEXT,
GetRscString(rgTips[lParam].residPrompt));
2543:            SetProp(hwndDlg, TIP_ID_PROP, (HANDLE)lParam);
2544:            break;
2545:    }
2546:
2547:    return(FALSE);
2548:}
2549:
2550:
2551://-------------------------------------------------------------------------
------------
2552:// SYMBOLPRICE compare function
2553://-------------------------------------------------------------------------
------------
2554:SGN SPCompare(SYMBOLPRICE *psp1, SYMBOLPRICE *psp2)
2555:{
2556:    return ((SGN)lstrcmp(psp1->szSymbol, psp2->szSymbol));
2557:}
2558:
2559:
2560://-------------------------------------------------------------------------
------------
2561:// Add Watch Position
2562://-------------------------------------------------------------------------
------------
2563:BOOL AddWatchPosition(CAccount *pAccount, TCHAR *pszTicker,
2564:            CDate *pDate, double dblUnits, double dblPrice)
2565:{
2566:    CPosition               *pPosition = NULL;
2567:    CTransaction    *pTrans = NULL;
2568:    CURID                   curid = CUR_US;
2569:
2570:    assert(*pszTicker != 0);        // gotta have a ticker, even if it is "none"
2571:
2572:    pPosition = GET_DATA(pAccount->GetOCX())->AddPosition(pszTicker, TRUE,
pAccount->GetAccountNumber());
2573:
2574:    if (pPosition)
2575:    {
2576:            GET_ROAMING(pAccount->GetOCX())->AddPositionToBatch(pPosition);
2577:
2578:            // Add currency...
2579:            curid = InferCurIdFromTicker(pszTicker, TRUE);
2580:            pPosition->SetCurrencyID(curid);
2581:
2582:            // Make sure we get quotes for this rate in the future...
2583:            OCX(pAccount->GetOCX())->AddCurrencyRate(curid);
2584:
```

```
2585:           // Apply currency scaling
2586:           dblPrice *= g_rgCurrency[curid].dblScaling;
2587:
2588:           pTrans = pPosition->AddTransaction(TT_BUY, TO_MANUAL, pDate,
dblUnits, dblPrice, 0);
2589:
2590:           if (pTrans)
2591:
GET_ROAMING(pAccount->GetOCX())->AddTransactionToBatch(pTrans, TRUE);
2592:   }
2593:
2594:   return (pTrans != NULL);
2595:}
2596:
2597:
2598://-------------------------------------------------------------------------
------------
2599:// SetupWatchAccount
2600://
2601:// Create the initial positions in the model portfolio...
2602://-------------------------------------------------------------------------
------------
2603:BOOL SetupWatchAccount(SETUP_STRUCT *pSetup, CAccount *pAccount)
2604:{
2605:   CPricePlex      *pPrices = NULL;
2606:   SYMBOLPRICE     sp = {0};
2607:   SYMBOLPRICE     *pSPCur, *pSPMac;
2608:   CLISTATUS       status;
2609:   TCHAR           *pszTickers = pSetup->pAds->szSymbols;
2610:   TCHAR           szTicker[MAX_TICKER_NAME];
2611:   int                     iSymbols;
2612:   int                     iDefWatchQuant =
pSetup->pOCX->GetDefaultWatchQuant();
2613:   CDate           date;
2614:   SYSTEMTIME      st;
2615:   BOOL            fNoPrice = FALSE;
2616:
2617:   GetLocalTime(&st);
2618:   date.SetDate(&st);
2619:
2620:   while (pszTickers != NULL)
2621:   {
2622:           pszTickers = GetNextTerm(pszTickers, SYMBREAKPTR, TRUE, szTicker,
MAX_TICKER_NAME, NULL);
2623:
2624:           if (*szTicker != 0)
2625:           {
2626:                   if (pPrices == NULL)
2627:                           pPrices = new CPricePlex(5, 5, &SPCompare);
2628:
2629:                   lstrcpyn(sp.szSymbol, szTicker, MAX_TICKER_NAME);
2630:                   sp.dblPrice = 0;
2631:
2632:                   pPrices->AddUniqueItem(&sp);
2633:           }
2634:   }
2635:
2636:   if ((pPrices == NULL) || (pPrices->GetCount() == 0))
2637:           return(FALSE);
2638:
2639:   if (pSetup->pAds->fModel)
2640:   {
2641:           CDate   *pDate;
```

```
2642:
2643:           if (!SetDateToString(&date, pSetup->pAds->szMODate, NULL))
2644:                   return(FALSE);
2645:
2646:           // Get the closing price at the transaction date, if a valid date
has been entered.
2647:           // If the transaction date is today, use the current price instead.
2648:           if (date.Compare(*g_pDateToday) == 0)
2649:                   pDate = NULL;
2650:           else
2651:                   pDate = &date;
2652:
2653:           pSetup->pOCX->InitWosaStatusStruct(&status, qrsPortBrief);
2654:
2655:           LookupPrices(pPrices, pDate, pSetup->pOCX->m_spClientSite, &status);
2656:
2657:           iSymbols = pPrices->GetCount();
2658:
2659:           if (iSymbols == 0)
2660:                   return(FALSE);
2661:   }
2662:   else
2663:   {
2664:           pSetup->pAds->mo = MO_NONE;
2665:   }
2666:
2667:   // If roaming, a batch must already be started when this code runs.
2668:
2669:   switch (pSetup->pAds->mo)
2670:   {
2671:           case MO_NONE:
2672:                   // Just add the symbols - ordinary watch account
2673:                   FORPLEX(pSPCur, pSPMac, *pPrices)
2674:                   {
2675:                           AddWatchPosition(pAccount, pSPCur->szSymbol, &date,
iDefWatchQuant, 0);
2676:                   }
2677:                   break;
2678:
2679:           case MO_DOLLARS:
2680:                   // $10000 per symbol.
2681:                   FORPLEX(pSPCur, pSPMac, *pPrices)
2682:                   {
2683:                           double p = pSPCur->dblPrice;
2684:                           if (p!=0)
2685:                                   AddWatchPosition(pAccount, pSPCur->szSymbol,
&date, 10000/p, p);
2686:                           else
2687:                                   fNoPrice = TRUE;
2688:                   }
2689:                   break;
2690:
2691:           case MO_SHARES:
2692:                   // 100 shares each
2693:                   FORPLEX(pSPCur, pSPMac, *pPrices)
2694:                   {
2695:                           AddWatchPosition(pAccount, pSPCur->szSymbol, &date,
100, pSPCur->dblPrice);
2696:
2697:                           if (CloseEnough(pSPCur->dblPrice, 0, 0.000001))
2698:                                   fNoPrice = TRUE;
2699:                   }
2700:                   break;
```

```
2701:
2702:           case MO_TOTAL:
2703:                   // pSetup->pAds->dblMOTotal for entire account
2704:                   FORPLEX(pSPCur, pSPMac, *pPrices)
2705:                   {
2706:                           double p = pSPCur->dblPrice;
2707:                           if (p!=0)
2708:                                   AddWatchPosition(pAccount, pSPCur->szSymbol,
&date, pSetup->pAds->dblMOTotal/p/iSymbols, p);
2709:                           else
2710:                                   fNoPrice = TRUE;
2711:                   }
2712:                   break;
2713:
2714:           default:
2715:                   assert(FALSE);
2716:   }
2717:
2718:   GET_DATA(pSetup->pOCX)->PostAllTransactions(POST_SUMS);
2719:   GET_LIST(pSetup->pOCX)->Refresh(TRUE);
2720:   OCX(pSetup->pOCX)->LaunchUpdate();
2721:
2722:   if (fNoPrice)
2723:   {
2724:           TCHAR   szError[MAX_ERROR_STRING];
2725:           BOOL    fFirst = TRUE;
2726:
2727:           GetRscStringToBuffer(IDS_ERR_MODEL_PRICE, szError);
2728:
2729:           FORPLEX(pSPCur, pSPMac, *pPrices)
2730:           {
2731:                   if (CloseEnough(pSPCur->dblPrice, 0.0, 0.000001))
2732:                   {
2733:                           if (fFirst)
2734:                                   fFirst = FALSE;
2735:                           else
2736:                                   my_strcatn(szError, ", ", MAX_ERROR_STRING);
2737:
2738:                           my_strcatn(szError, pSPCur->szSymbol,
MAX_ERROR_STRING);
2739:                   }
2740:           }
2741:
2742:           my_strcatn(szError, ". ", MAX_ERROR_STRING);
2743:
2744:           if ((pSetup->pAds->mo == MO_DOLLARS) || (pSetup->pAds->mo ==
MO_TOTAL))
2745:           {
2746:                   my_strcatn(szError, GetRscString(IDS_ERR_MODEL_CALCULATION),
MAX_ERROR_STRING);
2747:           }
2748:
2749:           MessageBox(OCX(pSetup->pOCX)->GetInnerWindow(), szError,
GetRscString(IDS_ERROR), MB_OK | MB_ICONSTOP | MB_APPLMODAL);
2750:   }
2751:
2752:   // cleanup the plex
2753:   if (pPrices)
2754:           delete pPrices;
2755:
2756:   return(TRUE);
2757:}
2758:
```

```
2759://-----------------------------------------------------------------------------
2760:// Dialog proc for the new account wizard.
2761://-----------------------------------------------------------------------------
2762:void InitAccountWizardPSP(CPortfolioControl *pOCX, SETUP_STRUCT *pSetup,
2763:                                                              INVPROPSHEETPAGE
psp[SETUP_WIZARD_PROPSHEETS])
2764:{
2765:    my_memset(psp, 0, sizeof(INVPROPSHEETPAGE) * SETUP_WIZARD_PROPSHEETS);
2766:
2767:    // Setup Wizard
2768:    psp[SWP_ACCOUNT_FTUE].dwSize = sizeof(INVPROPSHEETPAGE);
2769:    psp[SWP_ACCOUNT_FTUE].hInstance = OCX(pOCX)->GetInstance();
2770:    psp[SWP_ACCOUNT_FTUE].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_FTUE);
2771:    psp[SWP_ACCOUNT_FTUE].pfnDlgProc = (DLGPROC)AccountWizFTUEDialogProc;
2772:    psp[SWP_ACCOUNT_FTUE].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2773:    psp[SWP_ACCOUNT_FTUE].lParam = (long)pSetup;
2774:
2775:    // New Account
2776:    psp[SWP_ACCOUNT_NEW].dwSize = sizeof(INVPROPSHEETPAGE);
2777:    psp[SWP_ACCOUNT_NEW].hInstance = OCX(pOCX)->GetInstance();
2778:    psp[SWP_ACCOUNT_NEW].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_NEW);
2779:    psp[SWP_ACCOUNT_NEW].pfnDlgProc = (DLGPROC)AccountWizNewDialogProc;
2780:    psp[SWP_ACCOUNT_NEW].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2781:    psp[SWP_ACCOUNT_NEW].lParam = (long)pSetup;
2782:
2783:    // Regular Accounts
2784:    psp[SWP_ACCOUNT_REGULAR].dwSize = sizeof(INVPROPSHEETPAGE);
2785:    psp[SWP_ACCOUNT_REGULAR].hInstance = OCX(pOCX)->GetInstance();
2786:    psp[SWP_ACCOUNT_REGULAR].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_REGULAR);
2787:    psp[SWP_ACCOUNT_REGULAR].pfnDlgProc = (DLGPROC)AccountWizRegularDialogProc;
2788:    psp[SWP_ACCOUNT_REGULAR].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2789:    psp[SWP_ACCOUNT_REGULAR].lParam = (long)pSetup;
2790:
2791:    // Watch Accounts
2792:    psp[SWP_ACCOUNT_WATCH].dwSize = sizeof(INVPROPSHEETPAGE);
2793:    psp[SWP_ACCOUNT_WATCH].hInstance = OCX(pOCX)->GetInstance();
2794:    psp[SWP_ACCOUNT_WATCH].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_WATCH);
2795:    psp[SWP_ACCOUNT_WATCH].pfnDlgProc = (DLGPROC)AccountWizWatchDialogProc;
2796:    psp[SWP_ACCOUNT_WATCH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2797:    psp[SWP_ACCOUNT_WATCH].lParam = (long)pSetup;
2798:
2799:    // Import Money, Quicken, Yahoo!, OFX
2800:    psp[SWP_ACCOUNT_EXISTING_IMPORT].dwSize = sizeof(INVPROPSHEETPAGE);
2801:    psp[SWP_ACCOUNT_EXISTING_IMPORT].hInstance = OCX(pOCX)->GetInstance();
2802:    psp[SWP_ACCOUNT_EXISTING_IMPORT].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_IMPORT_EXISTING);
2803:    psp[SWP_ACCOUNT_EXISTING_IMPORT].pfnDlgProc =
(DLGPROC)AccountWizImportExistingDialogProc;
2804:    psp[SWP_ACCOUNT_EXISTING_IMPORT].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2805:    psp[SWP_ACCOUNT_EXISTING_IMPORT].lParam = (long)pSetup;
2806:
2807:    // Import Web
2808:    psp[SWP_ACCOUNT_WEB].dwSize = sizeof(INVPROPSHEETPAGE);
2809:    psp[SWP_ACCOUNT_WEB].hInstance = OCX(pOCX)->GetInstance();
2810:    psp[SWP_ACCOUNT_WEB].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_WEB);
2811:    psp[SWP_ACCOUNT_WEB].pfnDlgProc = (DLGPROC)AccountWizWebDialogProc;
2812:    psp[SWP_ACCOUNT_WEB].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2813:    psp[SWP_ACCOUNT_WEB].lParam = (long)pSetup;
2814:
2815:    psp[SWP_ACCOUNT_WEB_LIST].dwSize = sizeof(INVPROPSHEETPAGE);
2816:    psp[SWP_ACCOUNT_WEB_LIST].hInstance = OCX(pOCX)->GetInstance();
```

```
2817:    psp[SWP_ACCOUNT_WEB_LIST].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_WEB_LIST);
2818:    psp[SWP_ACCOUNT_WEB_LIST].pfnDlgProc = (DLGPROC)AccountWizWebListDialogProc;
2819:    psp[SWP_ACCOUNT_WEB_LIST].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2820:    psp[SWP_ACCOUNT_WEB_LIST].lParam = (long)pSetup;
2821:
2822:    psp[SWP_ACCOUNT_WEB_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
2823:    psp[SWP_ACCOUNT_WEB_FINISH].hInstance = OCX(pOCX)->GetInstance();
2824:    psp[SWP_ACCOUNT_WEB_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_WEB_FINISH);
2825:    psp[SWP_ACCOUNT_WEB_FINISH].pfnDlgProc =
(DLGPROC)AccountWizWebFinishDialogProc;
2826:    psp[SWP_ACCOUNT_WEB_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2827:    psp[SWP_ACCOUNT_WEB_FINISH].lParam = (long)pSetup;
2828:
2829:    // Money Accounts
2830:    psp[SWP_ACCOUNT_MONEY_1].dwSize = sizeof(INVPROPSHEETPAGE);
2831:    psp[SWP_ACCOUNT_MONEY_1].hInstance = OCX(pOCX)->GetInstance();
2832:    psp[SWP_ACCOUNT_MONEY_1].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_1);
2833:    psp[SWP_ACCOUNT_MONEY_1].pfnDlgProc = (DLGPROC)AccountWizMoney1DialogProc;
2834:    psp[SWP_ACCOUNT_MONEY_1].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2835:    psp[SWP_ACCOUNT_MONEY_1].lParam = (long)pSetup;
2836:
2837:    psp[SWP_ACCOUNT_MONEY_IMPORT].dwSize = sizeof(INVPROPSHEETPAGE);
2838:    psp[SWP_ACCOUNT_MONEY_IMPORT].hInstance = OCX(pOCX)->GetInstance();
2839:    psp[SWP_ACCOUNT_MONEY_IMPORT].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_IMPORT);
2840:    psp[SWP_ACCOUNT_MONEY_IMPORT].pfnDlgProc =
(DLGPROC)AccountWizMoneyImportDialogProc;
2841:    psp[SWP_ACCOUNT_MONEY_IMPORT].pszTitle = 0;
2842:    psp[SWP_ACCOUNT_MONEY_IMPORT].dwFlags = PSP_USETITLE;
2843:    psp[SWP_ACCOUNT_MONEY_IMPORT].lParam = (long)pSetup;
2844:
2845:    psp[SWP_ACCOUNT_MONEY_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
2846:    psp[SWP_ACCOUNT_MONEY_FINISH].hInstance = OCX(pOCX)->GetInstance();
2847:    psp[SWP_ACCOUNT_MONEY_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_MONEY_FINISH);
2848:    psp[SWP_ACCOUNT_MONEY_FINISH].pfnDlgProc =
(DLGPROC)AccountWizMoneyFinishDialogProc;
2849:    psp[SWP_ACCOUNT_MONEY_FINISH].pszTitle = 0;
2850:    psp[SWP_ACCOUNT_MONEY_FINISH].lParam = (long)pSetup;
2851:
2852:    // Quicken Accounts
2853:    psp[SWP_ACCOUNT_QUICKEN].dwSize = sizeof(INVPROPSHEETPAGE);
2854:    psp[SWP_ACCOUNT_QUICKEN].hInstance = OCX(pOCX)->GetInstance();
2855:    psp[SWP_ACCOUNT_QUICKEN].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_QUICKEN);
2856:    psp[SWP_ACCOUNT_QUICKEN].pfnDlgProc = (DLGPROC)AccountWizQuickenDialogProc;
2857:    psp[SWP_ACCOUNT_QUICKEN].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2858:    psp[SWP_ACCOUNT_QUICKEN].lParam = (long)pSetup;
2859:
2860:    psp[SWP_ACCOUNT_QUICKEN_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
2861:    psp[SWP_ACCOUNT_QUICKEN_FINISH].hInstance = OCX(pOCX)->GetInstance();
2862:    psp[SWP_ACCOUNT_QUICKEN_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_QUICKEN_FINISH);
2863:    psp[SWP_ACCOUNT_QUICKEN_FINISH].pfnDlgProc =
(DLGPROC)AccountWizQuickenFinishDialogProc;
2864:    psp[SWP_ACCOUNT_QUICKEN_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
2865:    psp[SWP_ACCOUNT_QUICKEN_FINISH].lParam = (long)pSetup;
2866:
2867:    // Master Password
2868:    psp[SWP_ACCOUNT_MASTER_PASSWORD].dwSize = sizeof(INVPROPSHEETPAGE);
2869:    psp[SWP_ACCOUNT_MASTER_PASSWORD].hInstance = OCX(pOCX)->GetInstance();
2870:    psp[SWP_ACCOUNT_MASTER_PASSWORD].pszTemplate =
```

```
MAKEINTRESOURCE(IDD_ACCOUNT_MASTER_PASSWORD);
2871:    psp[SWP_ACCOUNT_MASTER_PASSWORD].pfnDlgProc =
(DLGPROC)AccountWizMasterPasswordDialogProc;
2872:    psp[SWP_ACCOUNT_MASTER_PASSWORD].pszTitle = 0;
2873:    psp[SWP_ACCOUNT_MASTER_PASSWORD].dwFlags = PSP_USETITLE;
2874:    psp[SWP_ACCOUNT_MASTER_PASSWORD].lParam = (long)pSetup;
2875:
2876:    // OFX Accounts
2877:    psp[SWP_ACCOUNT_OFX_1].dwSize = sizeof(INVPROPSHEETPAGE);
2878:    psp[SWP_ACCOUNT_OFX_1].hInstance = OCX(pOCX)->GetInstance();
2879:    psp[SWP_ACCOUNT_OFX_1].pszTemplate = MAKEINTRESOURCE(IDD_ACCOUNT_OFX_1);
2880:    psp[SWP_ACCOUNT_OFX_1].pfnDlgProc = (DLGPROC)AccountWizOFX1DialogProc;
2881:    psp[SWP_ACCOUNT_OFX_1].pszTitle = 0;
2882:    psp[SWP_ACCOUNT_OFX_1].lParam = (long)pSetup;
2883:
2884:    psp[SWP_ACCOUNT_OFX_NOGROUPING].dwSize = sizeof(INVPROPSHEETPAGE);
2885:    psp[SWP_ACCOUNT_OFX_NOGROUPING].hInstance = OCX(pOCX)->GetInstance();
2886:    psp[SWP_ACCOUNT_OFX_NOGROUPING].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_NOGROUPING);
2887:    psp[SWP_ACCOUNT_OFX_NOGROUPING].pfnDlgProc =
(DLGPROC)AccountWizOFXNoGroupingDialogProc;
2888:    psp[SWP_ACCOUNT_OFX_NOGROUPING].pszTitle = 0;
2889:    psp[SWP_ACCOUNT_OFX_NOGROUPING].dwFlags = PSP_USETITLE;
2890:    psp[SWP_ACCOUNT_OFX_NOGROUPING].lParam = (long)pSetup;
2891:
2892:    psp[SWP_ACCOUNT_OFX_GROUPING_1].dwSize = sizeof(INVPROPSHEETPAGE);
2893:    psp[SWP_ACCOUNT_OFX_GROUPING_1].hInstance = OCX(pOCX)->GetInstance();
2894:    psp[SWP_ACCOUNT_OFX_GROUPING_1].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_1);
2895:    psp[SWP_ACCOUNT_OFX_GROUPING_1].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping1DialogProc;
2896:    psp[SWP_ACCOUNT_OFX_GROUPING_1].pszTitle = 0;
2897:    psp[SWP_ACCOUNT_OFX_GROUPING_1].dwFlags = PSP_USETITLE;
2898:    psp[SWP_ACCOUNT_OFX_GROUPING_1].lParam = (long)pSetup;
2899:
2900:    psp[SWP_ACCOUNT_OFX_GROUPING_2].dwSize = sizeof(INVPROPSHEETPAGE);
2901:    psp[SWP_ACCOUNT_OFX_GROUPING_2].hInstance = OCX(pOCX)->GetInstance();
2902:    psp[SWP_ACCOUNT_OFX_GROUPING_2].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_2);
2903:    psp[SWP_ACCOUNT_OFX_GROUPING_2].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping2DialogProc;
2904:    psp[SWP_ACCOUNT_OFX_GROUPING_2].pszTitle = 0;
2905:    psp[SWP_ACCOUNT_OFX_GROUPING_2].dwFlags = PSP_USETITLE;
2906:    psp[SWP_ACCOUNT_OFX_GROUPING_2].lParam = (long)pSetup;
2907:
2908:    psp[SWP_ACCOUNT_OFX_GROUPING_3].dwSize = sizeof(INVPROPSHEETPAGE);
2909:    psp[SWP_ACCOUNT_OFX_GROUPING_3].hInstance = OCX(pOCX)->GetInstance();
2910:    psp[SWP_ACCOUNT_OFX_GROUPING_3].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_GROUPING_3);
2911:    psp[SWP_ACCOUNT_OFX_GROUPING_3].pfnDlgProc =
(DLGPROC)AccountWizOFXGrouping3DialogProc;
2912:    psp[SWP_ACCOUNT_OFX_GROUPING_3].pszTitle = 0;
2913:    psp[SWP_ACCOUNT_OFX_GROUPING_3].dwFlags = PSP_USETITLE;
2914:    psp[SWP_ACCOUNT_OFX_GROUPING_3].lParam = (long)pSetup;
2915:
2916:    // End
2917:    psp[SWP_ACCOUNT_OFX_FINISH].dwSize = sizeof(INVPROPSHEETPAGE);
2918:    psp[SWP_ACCOUNT_OFX_FINISH].hInstance = OCX(pOCX)->GetInstance();
2919:    psp[SWP_ACCOUNT_OFX_FINISH].pszTemplate =
MAKEINTRESOURCE(IDD_ACCOUNT_OFX_FINISH);
2920:    psp[SWP_ACCOUNT_OFX_FINISH].pfnDlgProc =
(DLGPROC)AccountWizOFXFinishDialogProc;
2921:    psp[SWP_ACCOUNT_OFX_FINISH].pszTitle = (LPCTSTR)IDS_SETUP_WIZARD;
```

```
2922:    psp[SWP_ACCOUNT_OFX_FINISH].lParam = (long)pSetup;
2923:}
2924:
2925://-------------------------------------------------------------------------
-----------
2926:// The beginnings of Investor's tip wizard...
2927://
2928:// Puts up a dialog at the end of manual account creation to inform the user
how to create
2929:// investments. The iWhichTip variable is an index into the rgTips array.
2930://-------------------------------------------------------------------------
-----------
2931:void DoNewAccountTip(CPortfolioControl *pOCX, int iWhichTip)
2932:{
2933:    HKEY    hKey;
2934:    HRESULT hr;
2935:    DWORD   dw;
2936:
2937:    hr = RegCreateKeyEx(REGHIVE_PM, GetRscString(IDS_PREF_PREFERENCES_REGISTRY),
2938:                        0, NULL, 0, KEY_ALL_ACCESS, NULL, &hKey, &dw);
2939:
2940:    if ((hr != ERROR_SUCCESS) || (PrefGetLogical(hKey,
rgTips[iWhichTip].residPrefTag, FALSE) == FALSE))
2941:    {
2942:            if (DialogBoxParam(_Module.GetModuleInstance(),
MAKEINTRESOURCE(IDD_TIP),
2943:                               pOCX->GetInnerWindow(), (DLGPROC)TipDialogProc,
(LPARAM)iWhichTip))
2944:            {
2945:                    PrefPutLogical(hKey, rgTips[iWhichTip].residPrefTag, TRUE);
2946:            }
2947:    }
2948:
2949:    RegCloseKey(hKey);
2950:}
2951:
2952://-------------------------------------------------------------------------
-----------
2953:// The new account wizard.
2954://
2955:// Handles new accounts linked to Money or OFX via a New Account Wizard
2956://-------------------------------------------------------------------------
-----------
2957:BOOL NewAccountWizard(WORD wNAWEntry, WORD wID, COFX *pOFX, void *pControl,
HWND hwndParent)
2958:{
2959:    BOOL                        fResult = FALSE;
2960:    BOOL                        fNeedAddPositionTip = FALSE;
2961:    int                                 iRet;
2962:    CPortfolioControl       *pOCX = (CPortfolioControl *)pControl;
2963:    CAccount                    *pNewAccount = NULL;
2964:    SETUP_STRUCT            setup;
2965:    INVPROPSHEETPAGE            psp[SETUP_WIZARD_PROPSHEETS];
2966:    PROPSHEETHEADER         psh;
2967:
2968:    if (pOCX->GetDataFileReadOnly() == TRUE)
2969:            return (FALSE);
2970:
2971:    if (pOCX->IsClosing() == TRUE)
2972:            return (FALSE);
2973:
2974:    assert(hwndParent);
2975:
```

```
2976:   my_memset(&setup, 0, sizeof(SETUP_STRUCT));
2977:   my_memset(&psh, 0, sizeof(PROPSHEETHEADER));
2978:
2979:   setup.swatSelect = SWAT_NONE;
2980:   setup.pOCX = pOCX;
2981:   setup.wNAWEntry = wNAWEntry;
2982:   setup.wNAWModifier = NAW_MODIFIER_NONE;
2983:   setup.wID = wID;
2984:
2985:   // Check if we are doing Setup wizard or New Account wizard
2986:   switch (wNAWEntry)
2987:   {
2988:   case NAW_FIRST_TIME:
2989:           psh.nStartPage = SWP_ACCOUNT_FTUE;
2990:           setup.iFileVer = pOCX->GetData()->GetFileVersion();
2991:           assert(FileHasEncryptedOFXData(pControl) == FALSE);
2992:           break;
2993:
2994:   case NAW_NEW_ACCOUNT:
2995:           psh.nStartPage = SWP_ACCOUNT_NEW;
2996:           break;
2997:
2998:   case NAW_IMPORT_ACCOUNT:
2999:           psh.nStartPage = SWP_ACCOUNT_EXISTING_IMPORT;
3000:           break;
3001:
3002:   case NAW_REIMPORT_ACCOUNT:
3003:   case NAW_UPDATE_ACCOUNT:
3004:           setup.swatSelect = SWAT_OFX;
3005:
3006:           assert(pOFX);
3007:
3008:           if ((pOFX == NULL) || (!pOFX->InitOFX(FALSE)))
3009:                   return(FALSE);
3010:
3011:           setup.pOFX = pOFX;
3012:           pOFX->AddRef();
3013:
3014:           {
3015:                   CAccount        *pAccount = pOFX->GetCurAccount();
3016:
3017:                   assert(pAccount);
3018:
3019:                   // Reset update date to cause a full history update
3020:                   if (pAccount->GetBrokerID() != NULL_BROKER)
3021:                   {
3022:                           pOFX->SetCurSessionToAccount(pAccount);
3023:
3024:                           setup.wNAWModifier = NAW_MODIFIER_EXISTING_ACCOUNT;
3025:
3026:                           if (wNAWEntry == NAW_UPDATE_ACCOUNT)
3027:                                   psh.nStartPage = SWP_ACCOUNT_OFX_GROUPING_1;
3028:                           else
3029:                                   psh.nStartPage = SWP_ACCOUNT_OFX_GROUPING_3;
3030:                   }
3031:                   else
3032:                   {
3033:                           setup.wNAWModifier = NAW_MODIFIER_NEW_ACCOUNT;
3034:                           if (FileHasEncryptedOFXData(pControl)
3035:                                   || (setup.pOCX->GetSecurityLevel() ==
SECURITY_HIGH))
3036:                           {
3037:                                   psh.nStartPage = SWP_ACCOUNT_OFX_1;
```

```
3038:                                   }
3039:                                   else
3040:                                   {
3041:                                           psh.nStartPage =
SWP_ACCOUNT_MASTER_PASSWORD;
3042:                                   }
3043:                           }
3044:                   }
3045:
3046:                   break;
3047:
3048:           default:
3049:                   break;
3050:           }
3051:
3052:           assert(hwndParent);
3053:           psh.dwSize = PROPSHEETHEADER_V1_SIZE;
3054:           psh.dwFlags = PSH_PROPSHEETPAGE | PSH_WIZARD | PSH_NOAPPLYNOW;
3055:           psh.hwndParent = hwndParent;
3056:           psh.hInstance = OCX(pOCX)->GetInstance();
3057:           psh.pszIcon = NULL;
3058:           psh.nPages = sizeof(psp)/sizeof(INVPROPSHEETPAGE);
3059:
3060:           InitAccountWizardPSP(pOCX, &setup, psp);
3061:           psh.ppsp = (PROPSHEETPAGE*)(&psp);
3062:
3063:           OCX(pOCX)->ModalDialog(TRUE);
3064:
3065:           // Run the Setup Wizard PropSheets
3066:           iRet = PropertySheet(&psh);
3067:           if (iRet == -1) // PropertySheet may actually fail
3068:           {
3069:                   DWORD   dwError;
3070:                   TCHAR   szErrorMessage[256];
3071:                   TCHAR   szDisplayMessage[256*2];
3072:                   dwError = GetLastError();
3073:                   FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, NULL, dwError, 0,
szErrorMessage, 256, NULL);
3074:                   my_strcpyn(szDisplayMessage, "There has been a system error while
creating a New Account. ", 256*2);
3075:                   my_strcatn(szDisplayMessage, szErrorMessage, 256*2);
3076:                   MessageBox(NULL, szDisplayMessage, "Error", MB_OK |
MB_ICONINFORMATION);
3077:           }
3078:
3079:           OCX(pOCX)->ModalDialog(FALSE);
3080:
3081:           if (setup.pOFXDownload)
3082:                   ReleaseInterface(setup.pOFXDownload);
3083:
3084:           if (iRet == TRUE)
3085:           {
3086:                   switch (setup.swatSelect)
3087:                   {
3088:                   case SWAT_REGULAR:
3089:                   case SWAT_WATCH:
3090:                           setup.pAds->ulCodes |= (DLG_OK | DLG_REFRESH);
3091:
3092:                           // AddAccount will prompt if there are too many portfolios
3093:                           setup.pAds->pData =
GET_DATA(pOCX)->AddAccount(GET_DATA(pOCX)->GetAccountCount(),
3094:
setup.pAds->szAccountName,
```

```
3095:
setup.pAds->fWatch,
3096:
setup.pAds->aoOrigin,
3097:
setup.pAds->fCash);
3098:
3099:                   pNewAccount = setup.pAds->pData ?
setup.pAds->pData->CastToAccount() : 0;
3100:
3101:                   if (pNewAccount)
3102:                   {
3103:                           GET_ROAMING(pOCX)->StartBatch();
3104:
3105:                           GET_ROAMING(pOCX)->AddAccountToBatch(pNewAccount);
3106:
3107:                           switch (setup.swatSelect)
3108:                           {
3109:                           case SWAT_REGULAR:
3110:                                   // Create the cash account if needed
3111:                                   if (setup.pAds->fCash)
3112:
pNewAccount->CreateCashPosition(setup.pAds->dblCash);
3113:
3114:                                   if (!setup.pAds->fWatch)
3115:                                           setup.fLaunchBuyDialog = TRUE;
3116:
3117:                                   fNeedAddPositionTip = TRUE;
3118:                                   break;
3119:
3120:                           case SWAT_WATCH:
3121:                                   SetupWatchAccount(&setup, pNewAccount);
3122:                                   break;
3123:                           }
3124:
3125:
GET_ROAMING(pOCX)->UpdateAccountInBatch(pNewAccount);
3126:
3127:                           GET_ROAMING(pOCX)->EndBatch();
3128:
3129:                           fResult = TRUE; // Created an account
3130:                   }
3131:                   break;
3132:
3133:           case SWAT_YAHOO:
3134:           case SWAT_QUICKEN_COM:
3135:                   setup.pWeb->SetHWndParent(hwndParent);
3136:                   fResult = setup.pWeb->ImportData(TRUE);
3137:
3138:                   break;
3139:
3140:           case SWAT_QUICKEN:
3141:                   if (setup.pQuicken)
3142:                   {
3143:                           setup.pQuicken->SetHWndParent(hwndParent);
3144:                           fResult = setup.pQuicken->ImportData(TRUE);
3145:                   }
3146:                   break;
3147:
3148:           case SWAT_MONEY_IMPORT:
3149:           case SWAT_MONEY_LINK:
3150:                   if (setup.pMoney)
3151:                   {
```

```
3152:                           setup.pMoney->SetHWndParent(hwndParent);
3153:                           fResult = setup.pMoney->ImportData(TRUE);
3154:                   }
3155:                   break;
3156:
3157:           case SWAT_OFX:
3158:                   if (setup.pOFX)
3159:                   {
3160:                           setup.pOFX->SetHWndParent(hwndParent);
3161:                           fResult = setup.pOFX->ImportData(TRUE);
3162:                   }
3163:                   break;
3164:
3165:           default:
3166:                   break;
3167:           }
3168:
3169:           // We need to get historical prices on the securities we imported.
3170:           GET_DATA(pOCX)->SetHaveHistoricalPrices(FALSE);
3171:   }
3172:   else if (wNAWEntry == NAW_FIRST_TIME)
3173:   {
3174:           // user didn't create an account. Display a tip to tell him how to
later.
3175:           DoNewAccountTip(OCX(pControl), TIP_ACCOUNT);
3176:   }
3177:
3178:   //Destroy all unused objects
3179:   DeleteInterface(setup.pAds);
3180:   ReleaseInterface(setup.pQuicken);
3181:   ReleaseInterface(setup.pMoney);
3182:   ReleaseInterface(setup.pOFX);
3183:   ReleaseInterface(setup.pWeb);
3184:
3185:   // All pointers should be freed here.
3186:   assert(!setup.pOFX && !setup.pMoney && !setup.pQuicken && !setup.pAds &&
!setup.pWeb);
3187:
3188:   if (setup.fLaunchBuyDialog && pNewAccount)
3189:   {
3190:           CPortList *pList = GET_LIST(pOCX);
3191:
3192:           // Turn on display of watch accounts if they just created one and
want to add something to it
3193:           if (!pList->GetShowWatch() && pNewAccount->IsWatch())
3194:                   pList->SetShowWatch(TRUE);
3195:
3196:           if ((pList->GetFilter()) >= FIRST_ACCOUNT_VIEW)
3197:                   pList->SetFilter(pNewAccount->GetAccountNumber());
3198:
3199:           // Update to get the account added to the listview and give the user
feedback that
3200:           // something happened.
3201:           pList->Refresh(TRUE);
3202:           int iRow = pList->FindDataRecord(pNewAccount);
3203:           if (iRow != -1)
3204:                   pList->SelectItem(iRow, TRUE);
3205:
3206:           pOCX->BuyDialog(TT_BUY);
3207:
3208:           // add mDisplay a tip to tell him more investments later.
3209:           DoNewAccountTip(OCX(pControl), TIP_INVESTMENT);
3210:   }
```

```
3211:
3212:    return (iRet == TRUE);
3213:}
3214:
3215:
```

EXHIBIT C

Copy of "setupwiz.h" Computer Code

```
#ifndef SETUPWIZ_INCLUDED
#define SETUPWIZ_INCLUDED

#include "acctdlg.h"
#include "import.h"
#include "quicken.h"
#include "money.h"
#include "web.h"
#include "ofx.h"
#include "download.h"

#define TIP_ID_PROP                                    "TIPID"
#define SETUPPROP_PROP_NAME                          "SETUPPROP"
#define TIP_PROP_NAME                                "TIP"

typedef struct _NAW_TIP_STUCT
{
        RESID   residPrompt;
        RESID   residPrefTag;
        RESID   residBitmap;

} NAW_TIP_STUCT;

enum TIPS {
        TIP_ACCOUNT = 0,
        TIP_INVESTMENT,
        TIP_ROAMING
        };


enum SWP_ID {
        SWP_FIRST = 0,
        SWP_ACCOUNT_FTUE = SWP_FIRST,
        SWP_ACCOUNT_NEW,
        SWP_ACCOUNT_EXISTING_IMPORT,
        SWP_ACCOUNT_REGULAR,
        SWP_ACCOUNT_WATCH,
        SWP_ACCOUNT_WEB,
        SWP_ACCOUNT_WEB_LIST,
        SWP_ACCOUNT_WEB_FINISH,
        SWP_ACCOUNT_MONEY_1,
        SWP_ACCOUNT_MONEY_IMPORT,
        SWP_ACCOUNT_MONEY_FINISH,
        SWP_ACCOUNT_QUICKEN,
        SWP_ACCOUNT_QUICKEN_FINISH,
        SWP_ACCOUNT_MASTER_PASSWORD,
        SWP_ACCOUNT_OFX_1,
        SWP_ACCOUNT_OFX_NOGROUPING,
        SWP_ACCOUNT_OFX_GROUPING_1,
        SWP_ACCOUNT_OFX_GROUPING_2,
        SWP_ACCOUNT_OFX_GROUPING_3,
        SWP_ACCOUNT_OFX_FINISH,
        SWP_LAST = SWP_ACCOUNT_OFX_FINISH,
        SETUP_WIZARD_PROPSHEETS
        };

enum SW_ACCT_TYPE {
        SWAT_NONE = 0,
        SWAT_REGULAR,
        SWAT_WATCH,
        SWAT_IMPORT,
        SWAT_OFX,
        SWAT_MONEY,
```

```
        SWAT_MONEY_IMPORT,
        SWAT_MONEY_LINK,
        SWAT_QUICKEN,
        SWAT_YAHOO,
        SWAT_QUICKEN_COM
        };

// New account wizard entry points
enum NAWEntry {
        NAW_FIRST_TIME = 0,
        NAW_NEW_ACCOUNT,
        NAW_IMPORT_ACCOUNT,
        NAW_UPDATE_ACCOUNT,
        NAW_REIMPORT_ACCOUNT,
        };

// New account wizard entry point modifiers
#define NAW_MODIFIER_NONE                       0x0000
#define NAW_MODIFIER_NEW_ACCOUNT                0x0100
#define NAW_MODIFIER_EXISTING_ACCOUNT   0x0200

// Structure for getting account setup information
typedef struct _SETUP_STRUCT
{
        // Flow control variables
        WORD                            wNAWEntry;                      // Where we
started the wizard
        WORD                            wNAWModifier;           // Things that
affect the place we start
        WORD                            wID;                            //
Initiating command

        SW_ACCT_TYPE            swatSelect;                     // what kind of
account to add

        // Setup Wizard data
        int                                     iFileVer;                       //
Version of data loaded

        // Regular Acct data
        ACCTDLG_STRUCT          *pAds;
        BOOL                            fLaunchBuyDialog;       // launch "Record a
Buy" dialog when we're done?

        // Money Acct data
        BOOL                            fClosing;                       // TRUE if
we are editing and return should not mean next
        BOOL                            fNext;                          // TRUE if
NEXT button is enabled.
        CMoney                          *pMoney;

        // Quicken Acct data
        CQuicken                        *pQuicken;                      // Quicken
import object.

        // Web Acct data
        CWeb                            *pWeb;                          // Web
import object

        // OFX Acct data
        COFX                            *pOFX;                          // OFX
import object.
        CDownload                       *pOFXDownload;          // Current broker
```

```
profile download.

        // Master Password
        CPortfolioControl       *pOCX;
        CHAR                            szMasterPass[MAX_STRING];

} SETUP_STRUCT;

#define nDlgBrandLeft           350
#define nDlgBrandTop            17
#define nDlgBrandWidth          84
#define nDlgBrandHeight         34


BOOL AddWatchPosition(CAccount *pAccount, TCHAR *pszTicker, CDate *pDate, double
dblUnits, double dblPrice);

void DoNewAccountTip(CPortfolioControl *pOCX, int iWhichTip);

BOOL NewAccountWizard(WORD wNAWEntry, WORD wID, COFX *pAccount, void *pControl, HWND
hwndParent);


#endif
```